DIGITAL CONTROL OF A CONTACTLESS BATTERY CHARGING SYSTEM

by

AARON M. SCHULTZ

B.S., Electrical Engineering and Computer Engineering
Carnegie Mellon University
(May 1993)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

at the

Massachusetts Institute of Technology

February 1995

Signature of Author _____
Department of Electrical Engineering and Computer Science
February 6, 1995

Certified by _____
Steven B. Leeb
Carl Richard Soderberg Assistant Professor of Power Engineering
Thesis Supervisor

Accepted by _____
F.R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Digital Control of a Contactless Battery Charging System

by

Aaron M. Schultz

Submitted to the Department of Electrical Engineering and Computer Science

on February 6, 1995, in partial fulfillment of the

requirements for the degree of

Master of Science in Electrical Engineering

Abstract

This project develops and implements a safe, reliable method for contactless charging of electric vehicle batteries. The charging system's interface to the electric utility operates with unity power factor. This system accommodates many battery types by using a novel, large signal linear digital controller that can track arbitrary current profiles. Non-ohmic charging occurs through a two part, high frequency transformer. The primary side of the charging transformer remains outside the vehicle, and mates with a secondary installed in the vehicle. The scheme for impressing an AC signal on the primary of the charging connector involves lossless MOSFET switching in a high frequency inverter, and operates in the presence of high leakage inductance. Capacitive coupling enables contactless feedback of information from the battery in the secondary to the unity power factor charging supply in the primary.

Thesis Supervisor: Steven B. Leeb
Title: Carl Richard Soderberg Assistant Professor of Power Engineering

For my mother, my father, and my sister Lauren.

ACKNOWLEDGMENTS

I wish to acknowledge my advisor, Professor Steven B. Leeb, whose endless energy, intellectual curiosity, and unwavering tenacity sparked my continuous interest in this project. His seemingly countless areas of knowledge, in combination with the many-faceted nature of the research, enabled me to exercise and grow in many technical directions. The balanced connection between theory and practice which Professor Leeb espouses, as well as the strong desire for substantial results and quality documentation which he maintains, mesh well with my own outlook on the intellectual job of engineering.

I wish to thank Amp, Incorporated, who sponsored the research.

I wish to thank Vivian Mizuno, without whom it seems that LEES would collapse in a day. I also wish to thank the LEES staff for creating a friendly, enriching environment.

I salute my colleagues Ahmed, Deron, Kamakshi, Karen, Mary, Melissa, Rob, Steve, Tim, and Umair. I also hail the excellent folk Chris, Doug, Ethan, Ewa, Jan, Jay, Joe, Ken, Markus, Michael, Paul, Paul, Scott, friends and professors from CMU, and friends from home. And Jeanie.

I thank my father and mother, whose interest and support have never waned. And my beautiful sister, who is the most hard-working, cool sister one could have.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 Overview

The prevalence of electric vehicles may increase with the need for ecologically sound transportation. Potentially cleaner and quieter than combustion driven automobiles, electrically powered vehicles are expected, at minimum, to provide short distance transportation [10]. Electric vehicle battery charging stations must be efficient, safe, and flexible. The potential of multiply distributed charging stations, representing high demand from the electric utility, as well as limited battery storage capacity leading to frequent recharging [7], motivate efficiency in power supply construction. Safety must also affect design. Furthermore, a variety of available battery types will create the need for accommodating different battery charging profiles. It has been suggested that development of safe and cost effective contactless energy transfer will lead to the deployment of electric vehicles on public highways, and to the acceptance of electric vehicles by the general public [33].

This project researches a safe, reliable power electronic system for charging electric vehicle batteries. The charging system's interface to the electric utility operates with unity power factor to ensure maximally efficient power transfer from the source. Batteries have optimal charging currents [38]; this system accommodates many battery types by tracking arbitrary current profiles. Inductive coupling between the charging power supply and the battery provides an interface with no ohmic contact. An inductively coupled interface eliminates exposed contacts, reducing wear of sliding contacts and eliminating ohmic losses.

Three subsystems comprise the charger, as shown in Figure 1-1. First, a boost converter draws power from the electric utility and provides a regulated DC voltage bus. Since modern microcomputers are affordable and provide control flexibility in the field, this power supply is digitally controlled. The interface to the electric utility operates with unity power factor (UPF) to ensure maximum power transfer from the utility. Building on the large signal, linear controller presented in [34], this work develops a large signal, linear controller appropriate for battery charging applications. This controller enables the charger to track and deliver a wide range of desired charging current trajectories to many different types of batteries.

Second, a power electronic subsystem isolates the DC supply from the battery through inductive coupling. A high frequency inverter impresses an AC signal, developed from the DC power supply output, across the primary of a two part transformer. The primary side of the charging transformer remains outside the vehicle, and mates with a secondary installed in the vehicle. Energy flows through inductive coupling from the primary to the secondary. This arrangement enhances the safety and reliability of the charging system in comparison to chargers that employ conventional ohmic contacts. The high frequency transformer design is motivated by the inverse relationship between transformer size and signal frequency for a given power load. For maximum efficiency, a switching scheme uses parasitic elements of the power electronic devices to recover energy that would be otherwise dissipated [35] [36].

Third, a contactless system with capacitive coupling transmits feedback signals representing the output voltage and output current from the battery load in the vehicle to the power supply control system outside the vehicle. Steven Shaw designed and implemented this communication system [43].

1.2 Unity Power Factor

The power factor of a system is defined in [25, p. 396-9] as the ratio of average power delivered to the system to the product of the rms values of terminal voltage and current. Power factor is

$$k_p = \frac{< p(t) >}{V_{rms} I_{rms}} = \frac{< p(t) >}{S} \tag{1.1}$$

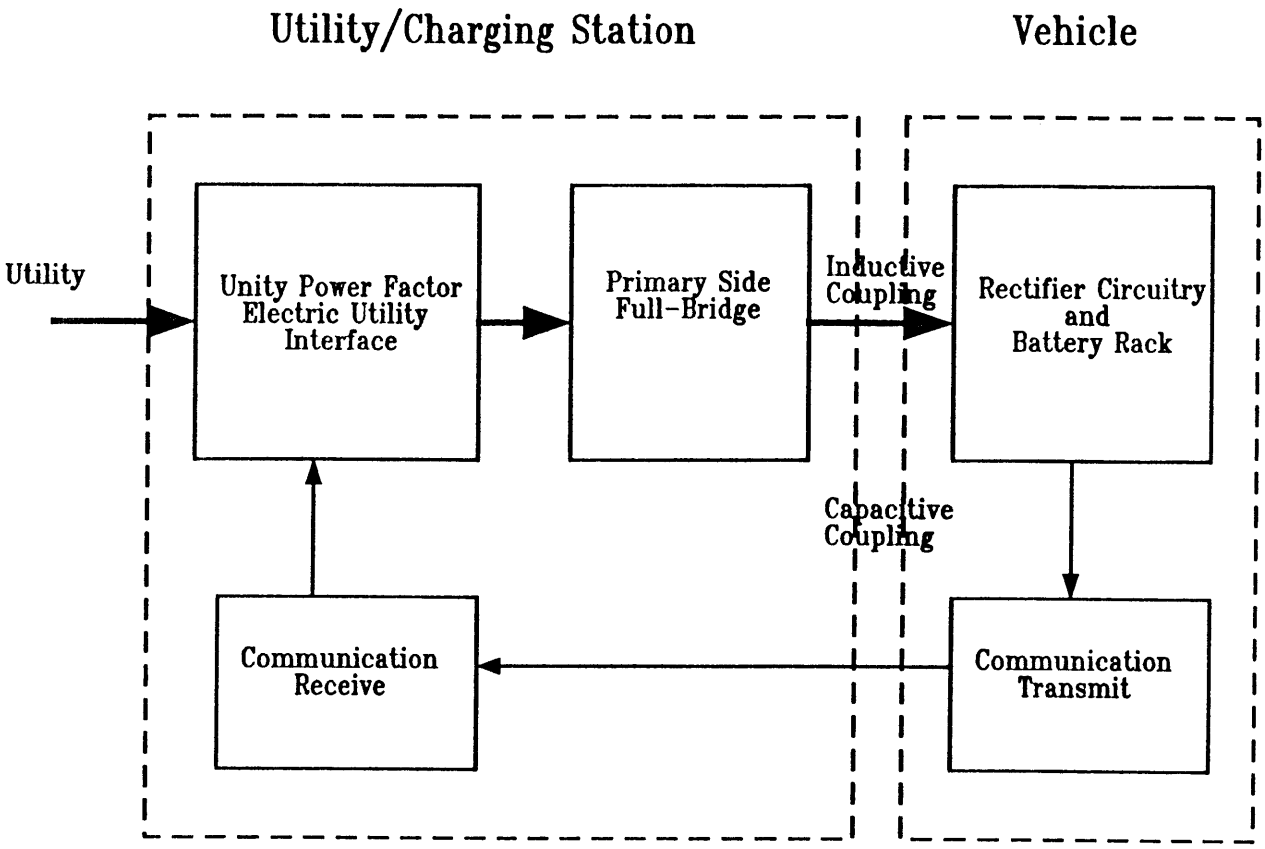

Figure 1-1: Charging System Overview

where $< p(t) >$ is the average power delivered and S is apparent power delivered. Average power is *real* power, i.e. power dissipated in the load. *Reactive* power is delivered by the source, but later returned to the source. When the power factor is unity, then the input current will be in phase with and exhibit the same wave shape, scaled in amplitude, as the input voltage.

Unity power factor operation at the input of a power supply provides at least two advantages. First, UPF operation maximizes the real work which can be performed given peak current limitations from a utility service. For such a limitation, the average power given by the integral of voltage and current over a period is maximized when the voltage and current are in phase, with the same wave shapes. In the case of battery chargers, maximizing the real work capability helps ensure that the battery charge as quickly as possible. Second, UPF operation minimizes harmonic currents that would generate harmonic voltages across the impedances of the utility, distorting the voltage waveform. When the power factor is exactly 1, then in fact the current waveform contains no higher harmonics, and the voltage waveform will be completely undistorted. A typical uncorrected input current to the full wave rectifier of a DC power supply, as shown in Figure 1-2, contains multiple higher harmonics. A purely sinusoidal input current contains none.

High performance battery chargers that seek to charge a battery as swiftly as possible require the highly efficient power supply capabilities of unity power factor systems. A case study on the impact of electric vehicle charging on power requirements from the utility predicts that energy needs will remain high during peak (day) as well as off-peak (overnight) hours [42]. A different experiment which tested battery charging and management maintained that a unity power factor source was necessary to avoid large energy loss over long periods of battery operation [3]. While the detailed study of distribution of power and scheduling of battery charging are beyond the scope of this project, the necessity for high efficiency and unity power factor operation evident in these and other studies were important motivations in the design of this battery charger.

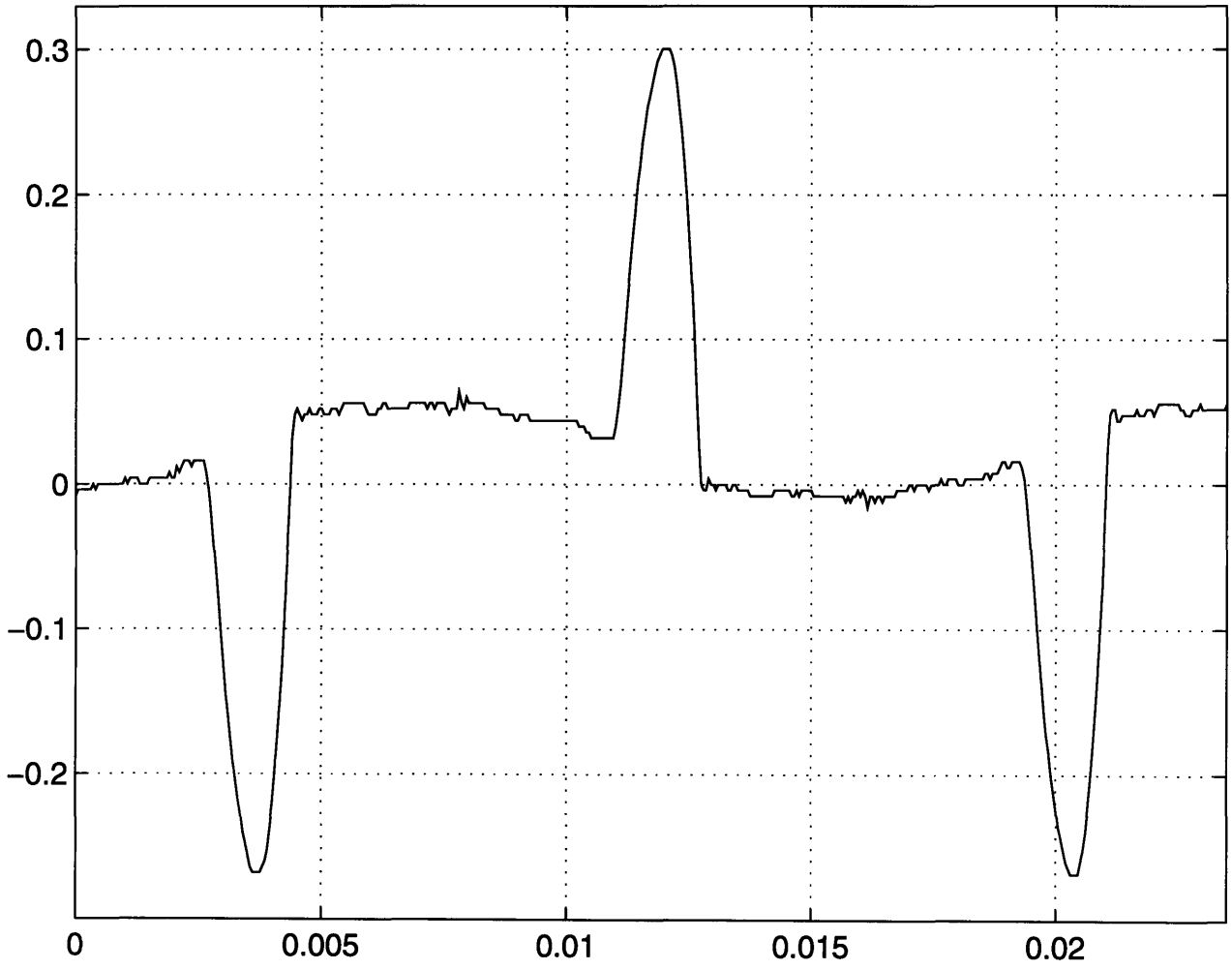

Figure 1-2: Uncorrected input current - one period.

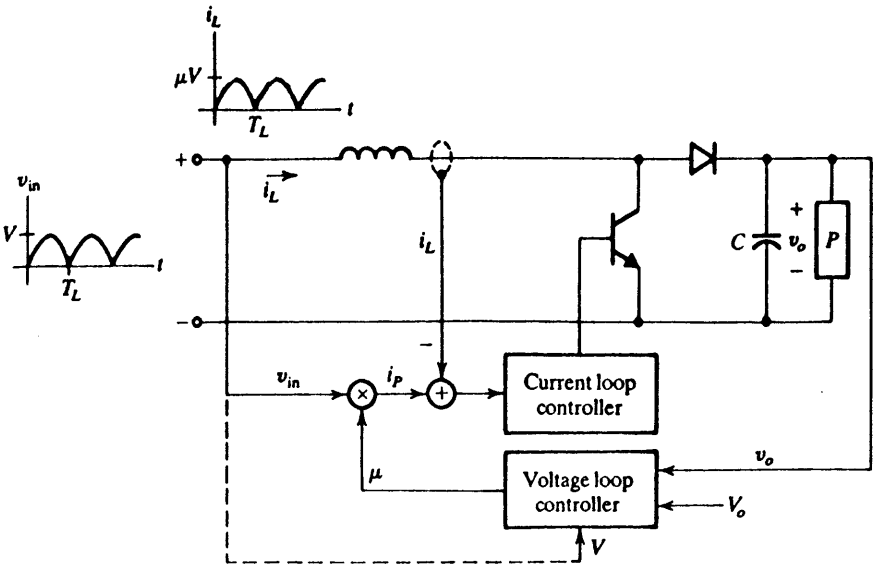

Figure 1-3: A High Power Factor AC/DC Switching Preregulator. Figure from J.G. Kassakian, M.F. Schlecht, G.C. Verghese, "*Principles of Power Electronics*", Addison-Wesley, 1991.

1.3 Boost Converter Topology

1.3.1 UPF Configuration

There are several commercially available methods for implementing power electronic power factor correction. A popular power converter topology for UPF converters is shown in Figure 1-3 [25, p. 397]. This topology is used in this project.

The boost converter input voltage is a rectified AC voltage waveform. The inner (current) control loop controls the source current to match the shape and phase of a reference waveform by providing a pulse width modulated switching sequence to the transistor that forces the inductor current $i_L(t)$ towards a desired current $i_p(t)$. The outer voltage loop computes a reference waveform $i_p(t)$ that is proportional to the input voltage ($i_p(t) = kv_{in}(t)$). The outer (voltage) loop regulates the output voltage v_o to the desired reference voltage V_o by adjusting the proportionality constant k used to generate i_p every line cycle. Changing k is tantamount to controlling input power, since input power for input current kV_{in} is $\frac{kV_{in}^2}{2}$.

1.3.2 Digital Control

Efforts have demonstrated that microcomputer control of battery charging is feasible and affordable [4] [46] [16]. Digital control can introduce tremendous flexibility. Rather than changing physical parts in the circuit or implementing complex analog adaptive hardware, different programs running on the same microcontroller can easy alter system dynamics. Computer programs can carry out calculations necessary for battery management that may be too complicated to implement with analog hardware. For example, Sanyo's popular SI-101 embedded microprocessor-controlled Fast Charge Control Module provides safety features, selectable charging functions, and battery parameter analysis [41]. Another example is the electric road vehicle $180V$ traction battery charger, controlled by an embedded microprocessor, and described in [14]. The SAB 80535 processor in this $10kW$ system chooses from among battery specific charge programs and charging profiles based on the vehicle to be charged.

The digital controller designed and implemented in [34] replaces the voltage loop controller shown in Fig. 1-3, and will be used in controlling the voltage loop for this project.

1.4 Modeling of UPF Power Supplies

In any control application, the physical system must be modeled for analytical evaluation unless complicated adaptive measures are taken. A plant such as a boost converter used as a UPF power supply is often modeled with a small signal transfer function operating around a nominal steady state point. This linearization leads to tractable linear equations, but can only work for applications where the circuit operates in the vicinity of a nominal or perhaps periodically varying operating point. Since the output current of the battery charger may need to vary over a large range, a large signal model must be developed. The input current / input voltage relationship $i_p(t) = kv_{in}(t)$ from UPF leads, as explained in [34], to a linear, large signal model for the power supply of Fig. 1-3. This model forms a basis for the current control presented in this work.

1.5 Inductively Coupled Interface

For safety reasons, inductively coupled power delivery has found applications in many settings. For example, a mining application which uses so-called coaxial-winding transformers was conceived and tested (50kW at 50kHz) [12]. The load is attached to a secondary winding to which power is magnetically coupled through ring cores. Another inductively coupled system uses two spiral coils to transfer tens of Watts of power at frequencies of $100 - 200kHz$ with 95% efficiency from a source outside the human body to an artificial heart [17].

The safety afforded by inductively coupled charging makes this approach attractive for use in electric vehicles. A system by Esser [13] charges electric vehicle batteries by using a core best described as two halves of a hollowed ring. This transformer has both an inductive channel for magnetically coupling power supply energy, and a capacitive channel for electrically coupling feedback signals. Esser's system achieves 5kW with up to 93% efficiency. Another project by the Pacific Gas and Electric Company, in conjunction with the Electric Power Research Institute and Inductran Corporation, installed a dual-inductor coupling system on a prototype G-Van [6]. This charger operated over 90% efficiently with high charging rates (3.5 hours from 0 to 100% capacity for a $162A - h$ battery).

One study stresses the importance of the contactless interface between the vehicle and the charger. Energy is transferred from an inductor in the floor or mounted onto a wall to an inductor inside a vehicle through an air gap [18]. The success of this charger demonstrates the feasibility of the similar magnetic coupling arrangement in the present work.

1.6 Modeling the Battery

The charging current profile and control scheme are dependent on the characteristics of the battery being charged. Many equivalent models have been proposed for a variety of battery types. Some models are based on physical modeling [29]. Maja and Spinelli's physical model, for example, takes into account mass and energy conservation, transport equations, and electro-chemical kinetics. Another effort by Copetti and Chenlo [9] starts with the simple I-V relationship for charging and discharging:

$$V = V_{oc} \pm RI. \tag{1.2}$$

Empirical studies and simplified modeling lead to equations for internal resistance, capacity, correction for temperature effects, and overcharging.

These studies represent theoretically, physically motivated descriptions which attempt to model battery mechanics. Both methods must rely, however, on empirical curves in finalizing their respective topologies, since completely accurate modeling of battery electrochemical activity has proven impossible.

Other investigations approach battery modeling from completely empirical perspectives. The studies by Jayne and Morgan [22] and by Blok, Horst, and Turkenburg [19] are further empirical studies. Various transients in voltage and current are imposed on the batteries, and variables such as output voltage, output current, temperature, and aging are measured. Jayne and Morgan identify differing responses to continuous charge / discharge versus pulsed charge / discharge. Blok, Horst, and Turkenburg try to justify a simply model

$$u \approx (e - cq_d) - ir(1 + k(\frac{q_{c,d}}{q_s - q_{c,d}})) \tag{1.3}$$

in which u is the normalized voltage, q_d is depth of discharge, q_c is the state of charge, and i is the normalized current. The parameters e, c, r, k, and q_s differ for charging and discharging.

An analysis by Mayer and Biscaglia [5] matches experimental data with

$$E = P_1 + P_2 Log(1 - \frac{Q}{C(I)}) \tag{1.4}$$

where P_1 is the voltage of the fully charged battery, P_2 is an empirical coefficient, Q is exchanged charge, and $C(I)$ is total capacity of the battery from the expression

$$C(I) = \frac{1}{aI^B + c} \tag{1.5}$$

with a, b, and c more empirical results.

All of these studies indicate the complexity involved in modeling a battery. Further complication is noted in a study by Gluck and Timmerman [15], who observe the effects

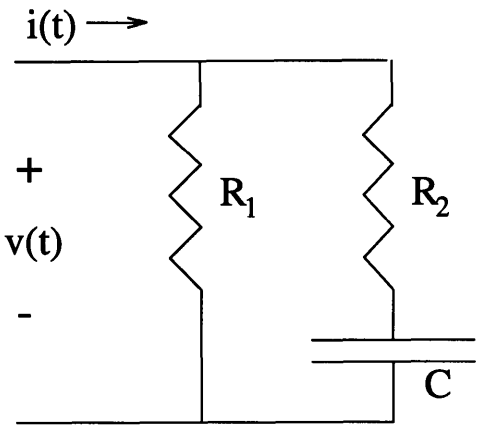

Figure 1-4: Simple Linear Battery Model

of a weak cell in a battery comprised of homogeneous cells. For the Ni-Cad batteries in question, polarization within the "bad" cell results in lower voltage across that cell, which places higher stress on the other cells as they try to maintain a net combined voltage.

One particularly simple battery model arises from a spacecraft electrical battery study [31]. This battery model has linear I-V characteristics, which is necessary in a linear, analytical description of the control loop. On an oscilloscope, McVey and Temkin measure battery voltage in response to charging and discharging currents. A simple RC network fits the I-V response, as shown in Figure 1-4.

1.7 Motivation

Of six alternative power sources for electric vehicles, namely electricity, reformulated gasoline, compressed natural gas, methanol, ethanol, and propane, a comparison concludes that in the United States, vehicles using electricity will have the best air quality benefit [8]. Electricity as a fuel is viewed as the most abundant, secure, safe, and suppliable alternative. The perceived inevitability of unacceptable pollution levels resulting from combustion engines has stimulated support for research and development of electric vehicles [26].

The goal of this project is to design and implement a prototype battery charger that operates with unity power factor while tracking a charging current profile. The charger will employ an inductively coupled power interface and a capacitively coupled interface for closed

loop feedback as in Figure 1-1.

1.8 Thesis Organization

This document is organized as follows. Chapter 2 discusses previous development and new enhancements of the voltage control of the system in Figure 1-3. The voltage controlled UPF system is then treated as the plant of a current control feedback system. The current control system is stabilized, and generalized for different loads.

Chapter 3 presents a DC/AC circuit which impresses an AC signal developed from the UPF DC power supply across the primary of a gapped core transformer. Using models of a transformer, inverter MOSFETs, and rectifier diodes, a resonant transition switching pattern is shown to effect virtually lossless DC/AC conversion. The leakage inductance of the transformer is considered in the switching scheme and in the design of the transformer.

Chapter 4 illustrates how the charging system is implemented. The hardware requirements for accomplishing UPF and digital control for the UPF system are outlined. Also presented are the flow of control in the software, as well as the scaling needed to represent the real world signals in the computer. The remainder of the chapter draws attention to the details of the transformer subsystem hardware. Circuits are explained for MOSFET gate switching patterns and drives. The specifications of the transformer that was built are listed.

Chapter 5 presents experimental results. The tests for the UPF current charger and inductive coupling systems are performed separately. The results are compared to simulations.

Chapter 6 summarizes the material presented and results obtained. Useful future work is outlined and other suggestions for improvement are given.

A step-by-step procedural listing on how to run the prototype, and the details of designing low pass analog filters are discussed in the appendices. The appendices also contain the capacitive coupling summary written by Steven Shaw [43], a complete set of schematics, and software listings.

Chapter 2

Modeling and Control Design

The goal of the controller developed for the battery charger is accurately to track charging current profiles. The DC power supply in Figure 1-1 accomplishes the charging current control with nested control loops. In this chapter, the dynamics of the inner voltage loop are presented first, since the behavior of the voltage loop directly affects the design of the complete closed loop current control system. A linear, discrete-time model of the UPF system is derived, and the voltage feedback loop of Figure 1-3 is compensated. Feed forward is employed to shield the voltage loop behavior from load dynamics. Finally, the current loop is stabilized based on a simplified model of the inner voltage loop.

2.1 Voltage Control

For applications involving regulation around an output specific operating point, nonlinear plant models may be linearized around the nominal operating point for control design. In tracking applications such as battery charging, the plant must be stabilized for large output deviations. An equation based on a power balance for the boost converter will be used to develop a large signal, linear controller [34] [25, p. 396-9].

2.1.1 Modeling the Boost Converter

The boost converter in Figure 1-3 can be modeled by a power balance equation. Equating the rate of change of stored energy in the capacitor to input power, rate of change of stored energy in the inductor, and power dissipated in the circuit elements, yields the equation:

$$\frac{1}{2}C\frac{dv_o^2}{dt} = v_{in}i_L - \frac{1}{2}L\frac{di_L^2}{dt} - P(t). \tag{2.1}$$

If the input current, also the inductor current $i_L(t)$, is taken to be $k(t)v_{in}(t)$, then equating the input power to the power dissipated in the circuit elements yields the equation

$$\frac{1}{2}C\frac{dv_o^2(t)}{dt} = k(t)v_{in}^2(t) - \frac{1}{2}L\frac{d[k^2(t)v_{in}^2(t)]}{dt} - P(t). \tag{2.2}$$

2.1.2 The T_L Averaged Model

The model in Eq. (2.2) is difficult to control because it is nonlinear (contains v_o^2 terms) and time-varying (coefficient of $k(t)$ is time-varying). Eq. (2.2) is linearized by treating the squared voltage v_o^2 rather than v_o as a variable. A large signal, linear, time-invariant model results from averaging the equation over a half line cycle T_L. The following development is taken from [28].

The running average of a variable is defined as

$$\bar{w}(t) = \frac{1}{T_L}\int_{t-T_L}^{t} w(\tau)d\tau. \tag{2.3}$$

The squared voltage variable $v_o^2(t)$ will be denoted by x. Assuming the variation in v_o^2 is small over a period T_L, then $x \approx \bar{v}_o^2$, and assuming that $k(t)$ remains constant over an interval of length T_L, then

$$\int_{t-T_L}^{t} k(\tau)v_{in}^2(\tau)d\tau = k(t)\frac{V^2}{2} \tag{2.4}$$

and

$$\int_{t-T_L}^{t} \frac{d[k^2(\tau)v_{in}^2(\tau)]}{d\tau}d\tau = 0 \tag{2.5}$$

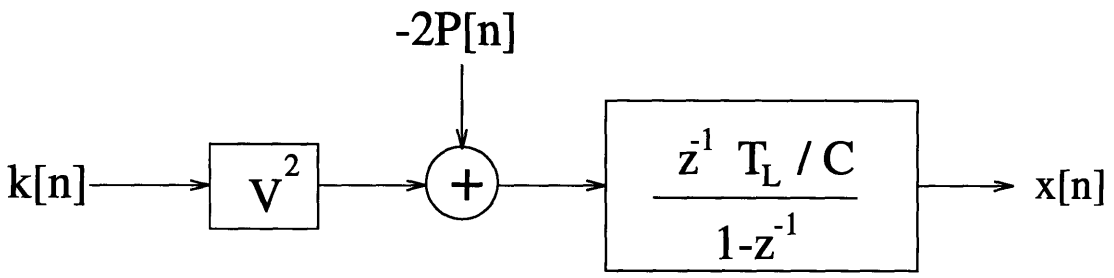

Figure 2-1: The sampled data model.

with V the peak of the input voltage. Since the output voltage and output current are assumed to vary slowly over one period T_L, then the output power $P(t)$ varies slowly over one period T_L. Substituting (2.4) and (2.5) into (2.2), with $P(t)$ not varying significantly over one period T_L, yields the linear, large signal, time-invariant first order description

$$\frac{dx(t)}{dt} = \frac{1}{C}(V^2 k(t) - 2P(t)). \tag{2.6}$$

The assumption that the ripples in $P(t)$ and v_o^2 do not vary greatly over the period T_L leads to "slow" control of the voltage system [23]. In this control scheme, $k(t)$ will be fixed for one period. Furthermore, since the boost converter capacitor is large, v_o^2 will not vary greatly over a period of $T_L = 8.3ms$. Thus (2.6) is a reasonable model from which to derive a linear, time-invariant squared output voltage controller.

2.1.3 The Sampled Data Model

For unity power factor, $k(t)$ must be kept constant over a line half-cycle. Then the phase and wave shape of the input current will be that of the input voltage for the duration of the half-cycle. Thus, a natural method of control for the voltage loop is to choose a new k once during each half-cycle. The time in one half-cycle is $\frac{1}{120Hz} = 8.3ms$. Performing the control calculations digitally will require all processing to be completed within $8.3ms$.

Integrating (2.6) over T_L yields the following sampled data equation [25, p. 396-9]:

$$\frac{1}{2}C(x[n+1] - x[n]) = T_L(k[n]\frac{V^2}{2} - P[n]) \tag{2.7}$$

or

$$x[n+1] = x[n] + \frac{T_L V^2}{C} k[n] - \frac{2T_L}{C} P[n], \tag{2.8}$$

where $x[n]$ and $P[n]$ are samples of $x(t)$ and $P(t)$ at times $t = nT_L$, and $k[n]$ is the peak input current to peak input voltage ratio during the interval between nT_L and $(n+1)T_L$. Eq. (2.8) is a linear, large signal discrete state-space representation of the dynamics of the boost converter. This equation models the sampled behavior of the output voltage *squared.* A z-transform block diagram of this model is shown in Figure 2-1, where the z-transform is taken from [24, p. 318].

2.1.4 PI Control

The sampled data model (2.8) suggests the use of discrete-time control to compensate the system. Calculating $k[n]$ as a linear function of $x[n]$, and inserting $k[n]$ back into (2.8) leads to linear control of $x[n]$. A computationally simple control scheme which eliminates steady-state error and stably compensates the system is *proportional / integral (PI)* control. This scheme is used to compensate the boost converter plant model of this battery charger [34] [25, p. 396-9]. PI control uses an accumulated error term

$$\sigma_v[n+1] = \sigma_v[n] + (X - x[n]) \tag{2.9}$$

with X the squared voltage reference and $(X - x[n]) = x_{err}$ the squared voltage error at time n. The control signal k is set equal to the sum of the products of a proportional gain G_P and v_{err}^2, and an accumulator gain G_I and σ_v:

$$k[n] = G_P x_{err}[n] + G_I \sigma_v[n]. \tag{2.10}$$

Defining the normalized gains

$$h_1 = \frac{T_L V^2}{C} G_P \tag{2.11}$$

$$h_2 = \frac{T_L V^2}{C} G_I \tag{2.12}$$

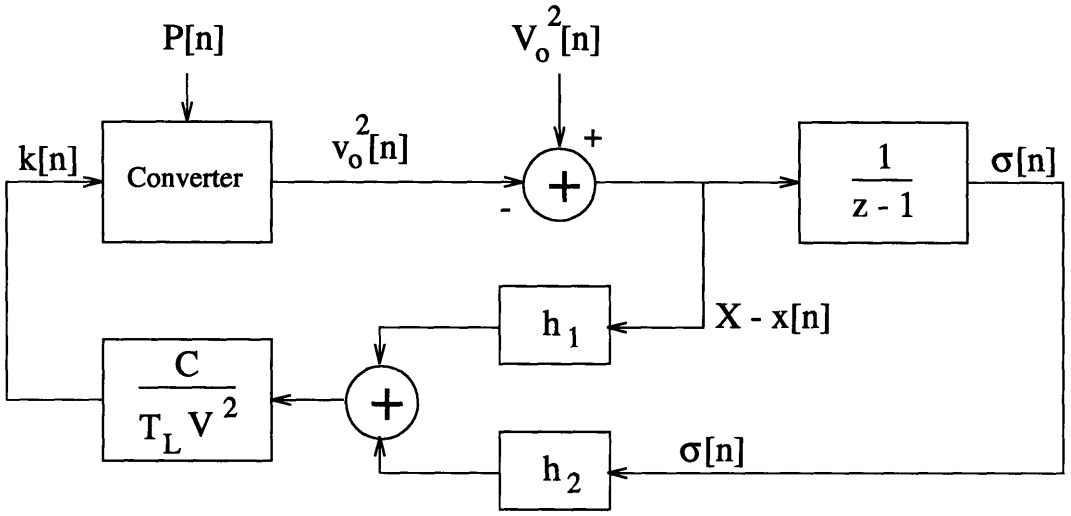

Figure 2-2: Closed loop discrete-time PI controller.

yields the command signal

$$k[n] = \frac{C}{T_L V^2}(h_1(X - x[n]) + h_2\sigma_v[n]). \tag{2.13}$$

The resulting closed-loop system is shown in Figure 2-2 [25, p. 396-9].

Combining (2.9) and (2.13) with (2.8) leads to the second order state space formulation for the system

$$\begin{bmatrix} \sigma_v[n+1] \\ x[n+1] \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ h_2 & (1-h_1) \end{bmatrix} \begin{bmatrix} \sigma_v[n] \\ x[n] \end{bmatrix} + \begin{bmatrix} 1 \\ h_1 \end{bmatrix} X[n] + \begin{bmatrix} 0 \\ -\frac{2T_L}{C} \end{bmatrix} P[n]. \tag{2.14}$$

The characteristic equation of the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ h_2 & (1-h_1) \end{bmatrix} \tag{2.15}$$

is found by setting the determinant of $z\mathbf{I} - \mathbf{A}$ to zero:

$$z^2 - (2-h_1)z + 1 - h_1 + h_2 = 0. \tag{2.16}$$

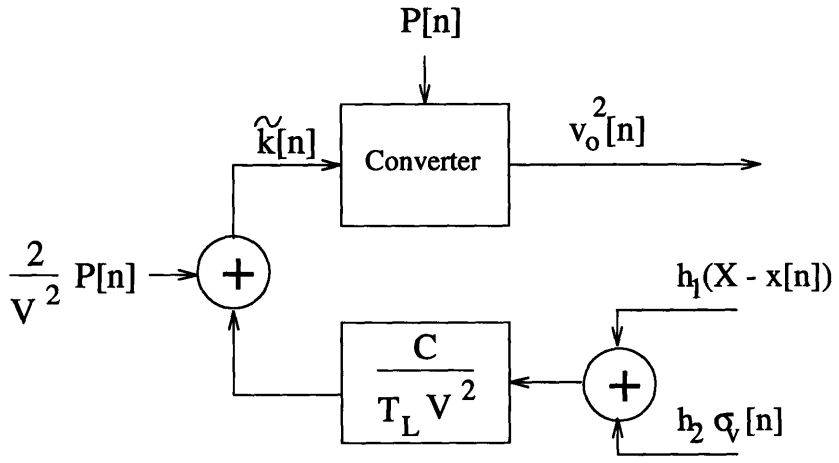

Figure 2-3: Feed forward in the control signal.

The roots of this equation, which are the system poles, are

$$z_1, z_2 = \frac{(2 - h_1) \pm \sqrt{h_1^2 - 4h_2}}{2}. \tag{2.17}$$

Setting these poles to have magnitude less than one results in a stable system, since the transient response of the the system has the form $c_1 z_1^k + c_2 z_2^k$, with $z_1 \neq z_2$ and c_1 and c_2 constants that depend on initial conditions.

2.1.5 Feed Forward Design

The matrix **A** in (2.15) does not represent the eigenvalues of the PI compensated closed voltage loop if the P term in (2.14) depends on the state variables σ_v and x. For boost converter loads more complicated than a resistor that add state to the system (Fig. 1-4, eg.), the two-state linear model of (2.14) will require more equations to represent the system. These new equations that describe the load dynamics need to be incorporated into the closed loop state space description. Because the load is linear in v_o, writing the power term P in terms of v_o^2 proves impossible.

Since including the load dynamics in the A matrix is impossible, a better idea is to isolate the closed loop system behavior from the load behavior. The method that accomplishes this separation is to add a feed forward term to the control signal k in (2.13). In this charging current control system, both the output voltage and the output current are available as

inputs to the controller, thus rendering the power available with no extra hardware. Figure 2-3 details the part of Figure 2-2 where the feed forward term is added. The new control signal is

$$\tilde{k}[n] = k[n] + \frac{2}{V^2}P[n], \tag{2.18}$$

or

$$\tilde{k}[n] = \frac{C}{T_L V^2}(h_1(X - x[n]) + h_2\sigma_v[n]) + \frac{2}{V^2}P[n]. \tag{2.19}$$

Substituting (2.19) into (2.8) yields a new second order linear model

$$\begin{bmatrix} \sigma_v[n+1] \\ x[n+1] \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ h_2 & (1-h_1) \end{bmatrix} \begin{bmatrix} \sigma_v[n] \\ x[n] \end{bmatrix} + \begin{bmatrix} 1 \\ h_1 \end{bmatrix} X[n] \tag{2.20}$$

which is independent of the load. Thus, once stable poles have been chosen and the PI gains have been calculated, the system will remain stable *for almost any load.* The stable voltage loop will converge to any reference given enough iterations. This guaranteed convergence for almost any load leads to a simple, linear model for the complete, closed loop voltage loop system whose input is a voltage reference and whose output is the boost converter output voltage.

2.2 Current Control

The goal of the charger is to track a current profile. Thus far only the inner voltage loop compensator that controls the unity power factor boost converter has been presented. This section introduces the mechanism for controlling the output current of the boost converter. A feedback loop is closed around the UPF voltage loop, as shown in Figure 2-4. The current controller provides the UPF system with a voltage reference.

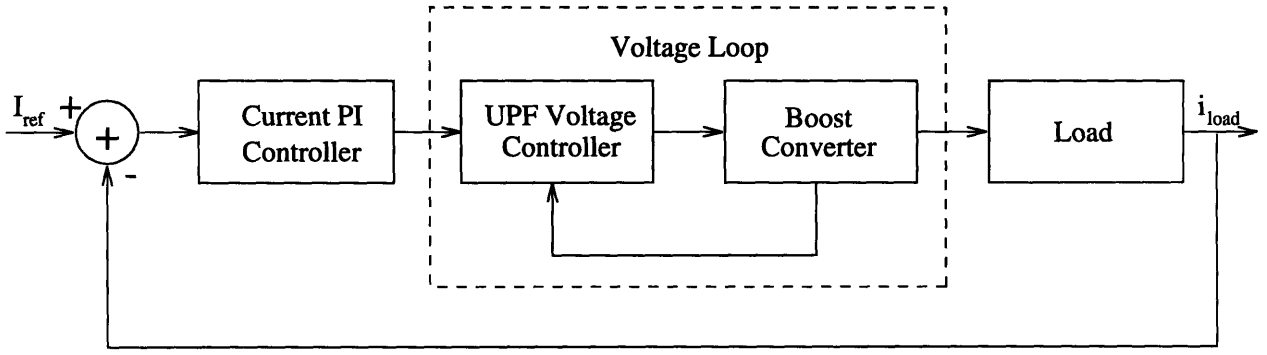

Figure 2-4: Closed loop current control.

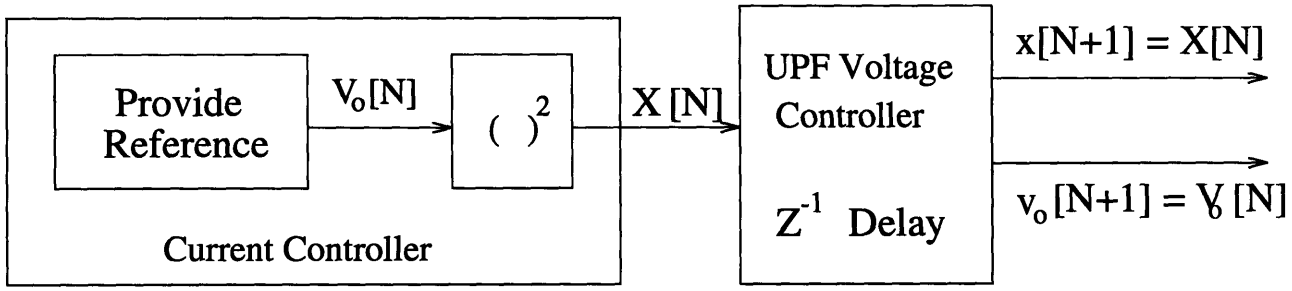

Figure 2-5: Current controller supplies squared reference for UPF voltage controller.

2.2.1 Time Scale Separation

From the analysis in Section 2.1.5, it is clear that the voltage loop can be stabilized for a wide range of loads. For a large enough number of increments of n, say Q, the voltage loop will converge. That is, $c_1 z_1^Q + c_2 z_2^Q$, the response of a state in the voltage system with $z_{1,2}$ the system eigenvalues, can be brought arbitrarily near zero for large enough Q. After Q steps, the output voltage is guaranteed to match the reference voltage.

The idea of operating the voltage loop much faster than the current loop has found usage in similar charging systems [23]. If the current loop operates at a sampled data rate of $N = Qn$, then when it supplies a voltage reference to the inner voltage loop at time $N = Qn$, it can expect that the output voltage will equal the reference Q steps of n later. Thus the current loop can assume the voltage loop is simply a delay (z-transform z^{-1}). As shown in Figure 2-5, the current loop supplies a squared output voltage reference at time N, and expects that the squared output voltage at time $N + 1$ is the reference. The current loop controls the unsquared voltage, but supplies a squared voltage reference to the UPF system.

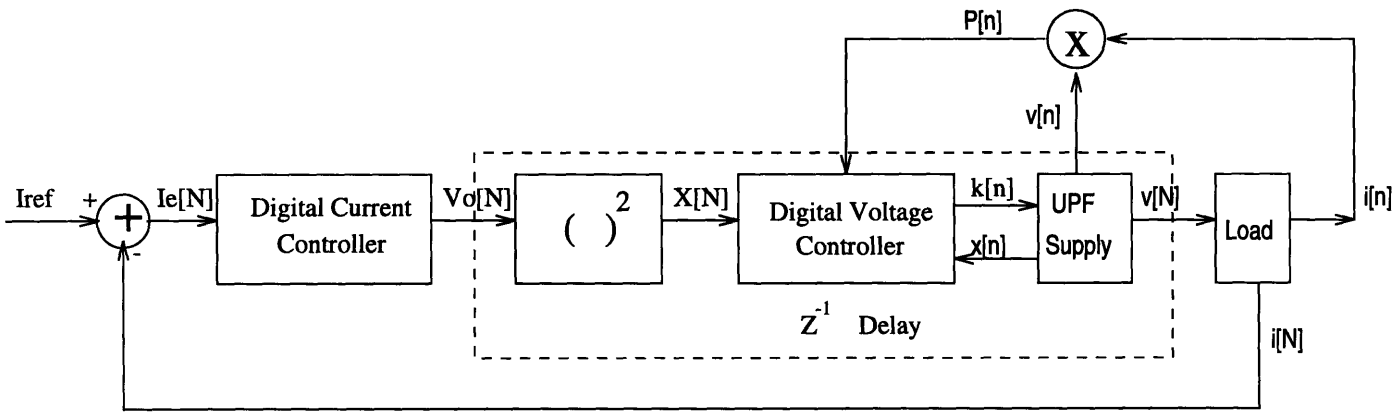

Figure 2-6: Entire Current Control System

2.2.2 PI Control

Examination of the closed loop pole locations for the linear, time-invariant plant and controller in Figure 2-4 indicates stability and transient performance. Linearity of all three blocks in Figure 2-4 ensures that linear control of the charging current is possible. Certainly, the delay model of the UPF voltage loop block is linear and time-invariant. Section 2.2.3 describes a method for deriving linear, sampled data models for certain battery loads (Fig. 1-4, eg.) that comprise the load block. This section evaluates the closed current loop poles using a PI compensator in the controller block that provides linear control to the loop. The entire linear charging current closed loop system is shown in Figure 2-6.

With PI control, the reference V_o is computed by

$$V_o[N] = h_3(I_{ref} - i[N]) + h_4\sigma_i[N] \tag{2.21}$$

where h_3 and h_4 are PI gains, i is the output current, I_{ref} is the current reference, and σ_i is the accumulator. The accumulator dynamics are described by

$$\sigma_i[N+1] = \sigma_i[n] + (I[N] - i[N]). \tag{2.22}$$

Applying the z^{-1} voltage loop model, the output voltage is equal to the delayed PI command signal, i.e.

$$v[N+1] \;=\; V_o[N] \;=\; h_3(I_{ref} - i[N]) \;+\; h_4\sigma_i[N]. \tag{2.23}$$

Assuming a simple resistive load R and the z^{-1} model for the voltage loop, then a state equation for i can be developed from Ohm's law

$$i[N+1] \;=\; \frac{v[N+1]}{R} \;=\; \frac{V[N]}{R}. \tag{2.24}$$

The state equation for i is

$$i[N+1] \;=\; \frac{h_3}{R}(I - i[N]) \;+\; \frac{h_4}{R}\sigma_i[N]. \tag{2.25}$$

Assembling (2.22) and (2.25) leads to the closed current loop state description

$$\begin{bmatrix} \sigma_i[N+1] \\ i[N+1] \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ \frac{h_4}{R} & \frac{-h_3}{R} \end{bmatrix} \begin{bmatrix} \sigma_i[N] \\ i[N] \end{bmatrix} + \begin{bmatrix} 1 \\ \frac{h_3}{R} \end{bmatrix} I[n]. \tag{2.26}$$

Setting the determinant of of $z\mathbf{I} - \mathbf{A}$ to zero yields the poles

$$z_1, z_2 \;=\; \frac{(1 - \frac{h_3}{R}) \pm \sqrt{(\frac{h_3}{R})^2 + \frac{2h_3}{R} + 1 - \frac{4h_4}{R}}}{2}. \tag{2.27}$$

2.2.3 Step-Invariant Transform

Since the battery charger is digitally controlled, one part of the system operates in continuous-time, while the other operates in discrete-time. The mix of discrete and continuous-time operation in the blocks of Figure 2-4 is shown in the diagram in Figure 2-7. Writing discrete-time equations that lead to closed loop pole placement for the system in Figure 2-7 requires sampled data models of the boost converter and the load.

The boost converter is already modeled by (2.8); the discrete-time model ($H(z)$) of the continuous-time load transfer function ($H(s)$) is obtained by considering how $H(s)$ is incorporated into the discrete-time current control loop. Figure 2-8 illustrates the interface between the discrete-time input and outputs to the load, and the continuous-time load behavior [44, p. 257]. The DT boost converter output voltage $v[N]$ is converted to CT by

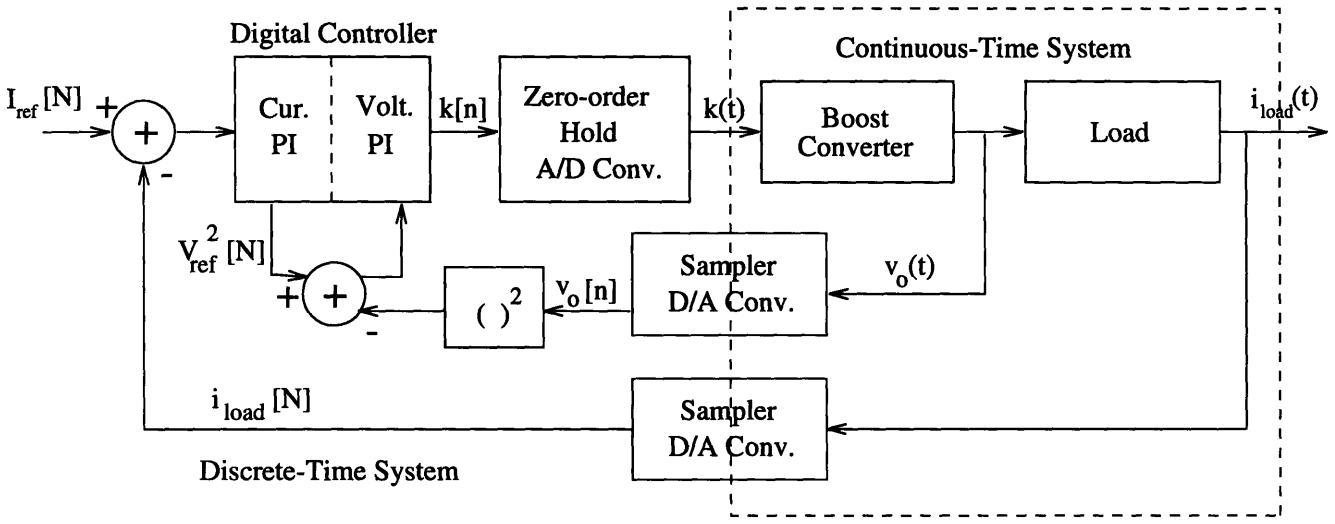

Figure 2-7: Closed current loop diagram illustrating the interface between discrete and continuous-time blocks.

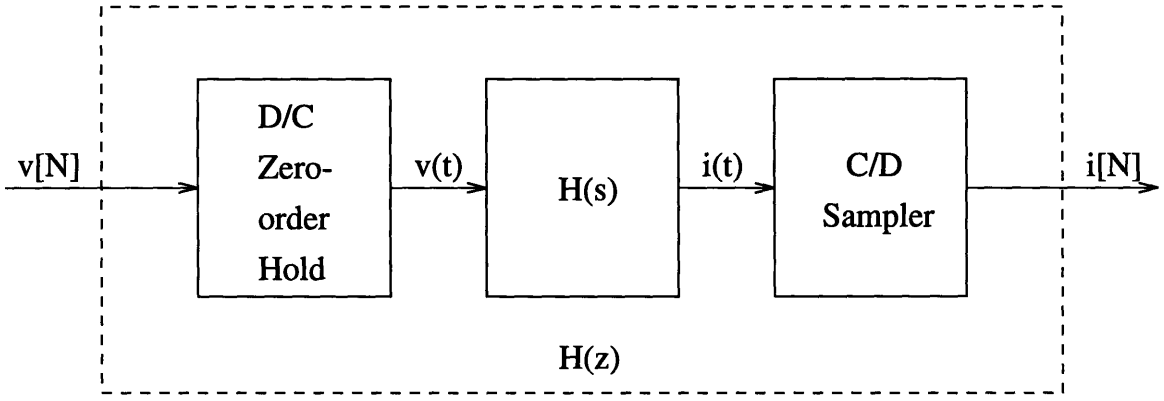

Figure 2-8: DT equivalent H(z) of CT transfer function H(s) interfaced into a DT system.

a zero-order hold operation. $H(s)$ operates on the continuous-time voltage $v(t)$ to produce the continuous-time current $i(t)$. Finally, the discrete-time boost converter output current $i[N]$ is obtained by sampling [44, p. 256].

The DT transfer function $H(z)$ can be found by applying a *step-invariant transform* to $H(s)$ [2, p. 54]. The step-invariant transform of a continuous-time system generates a discrete-time model whose step response is identical at time k to the continuous-time step response at time $t = NT$ [44, p. 256]. This particular CT-DT transformation is useful for this control system since the behavior of interest to the control loop is the voltage step response of the load. The step-invariant transform derives an $H(z)$ from $H(s)$ that is equivalent to

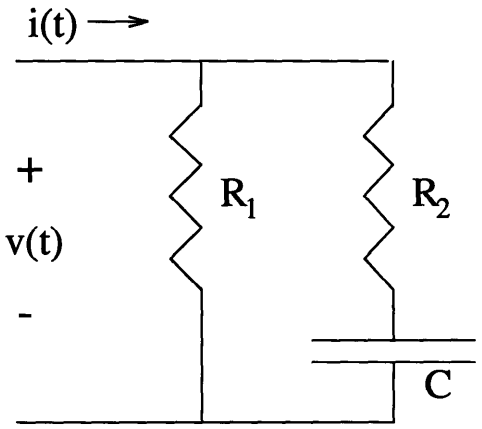

Figure 2-9: Simple Linear Battery Model

$H(z)$ in Figure 2-8 [2, p. 54].

The DT transfer function $H(z)$ is computed by the step-invariant transform as follows:

1. Compute the step response of $H(s)$. That is, calculate the inverse Laplace Transform of $\frac{H(s)}{s}$.

2. Sample the resulting continuous-time response $x(t)$ to obtain $x[N] = x(NT)$.

3. Determine the z-transform of $x[N]$, denoted by $X(z)$.

4. The z-transform $X(z)$ represents the step response of the DT transfer function $H(z)$, i.e. $\frac{zH(z)}{z-1}$. To find $H(z)$, multiply $X(z)$ by $\frac{z-1}{z}$.

For the load resistor R whose constitutive law is $V(t) = RI(t)$, the sampled data model is $V[N] = RI[N]$. More complicated load models require the use of the step-invariant transform.

As an example, consider the circuit from Figure 1-4, repeated for convenience in Figure 2-9. First, a continuous-time differential equation is derived using Kirchhoff's current and voltage laws and the device constitutive laws.

$$i(t) = \frac{v(t)}{R_1} + C\frac{dv(t)}{dt}(1 + \frac{R_2}{R_1}) - CR_2\frac{di(t)}{dt} \tag{2.28}$$

Assuming zero initial conditions, the Laplace transform yields the transfer function

$$\frac{I(s)}{V(s)} = \frac{\frac{1}{R_1} + sC(1 + \frac{R_2}{R_1})}{1 + sR_2C}. \tag{2.29}$$

The step response is $\frac{I(s)}{V(s)} * \frac{1}{s}$ with $V(s) = \frac{1}{s}$. Applying a partial fraction expansion and the inverse Laplace transform results in the continuous-time step response

$$i(t) = \frac{1}{R_1}u(t) + \frac{1}{R_2}e^{-\frac{t}{R_2C}}u(t). \tag{2.30}$$

Sampling this equation at $t = NT$ yields the discrete-time step response

$$i[N] = \frac{1}{R_1}u[N] + \frac{1}{R_2}e^{-\frac{NT}{R_2C}}u[N]. \tag{2.31}$$

The z-transform of (2.31) is in the form $I[z] = H[z] * S[z]$, where $H(z) = \frac{I[z]}{V[z]}$ and $S[z]$ is the z-transform of the step function, $z/(z-1)$. To find out the transfer function $I[z]/V[z]$, the z transform of (2.31) must be multiplied by $(z-1)/z$. The result is

$$I[z](z - e^{-\frac{T}{\tau}}) = V[z]\frac{1}{R_1}(z - e^{-\frac{T}{\tau}}) + V[z]\frac{1}{R_2}(z-1) \tag{2.32}$$

where $\tau = R_2C$.

As an illustration, Fig. 2-10 compares, for a particular set of component values, the continuous step response of (2.29) with the digital step response of (2.32). The dashed line is the CT response, and the dotted line is the DT response. The two curves are exactly the same at the times $t = NT$ at which both curves have points.

Recognizing that the inverse z-transform of $zX[z]$ is $x[N+1]$ or $x(NT+T)$, then taking the inverse z-transform of (2.32) yields

$$i[N+1] = e^{-\frac{T}{\tau}}I[N] + v[N+1](\frac{1}{R_1} + \frac{1}{R_2}) - v[N](\frac{e^{-\frac{T}{\tau}}}{R_1} + \frac{1}{R_2}). \tag{2.33}$$

Equation (2.33) is the discrete-time model for the load in Figure 2-9. Assembling (2.33), the PI equation (2.22), and the voltage delay equation (2.23), the state space model of the charging current closed loop system is

$$\begin{bmatrix} \sigma_i[N+1] \\ i[N+1] \\ v[N+1] \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ h_4(\frac{1}{R_1}+\frac{1}{R_2}) & (e^{-\frac{T}{\tau}} - h_3(\frac{1}{R_1}+\frac{1}{R_2})) & -(\frac{e^{-\frac{T}{\tau}}}{R_1}+\frac{1}{R_2}) \\ h_4 & -h_3 & 0 \end{bmatrix} \begin{bmatrix} \sigma_i[N] \\ i[N] \\ v[N] \end{bmatrix} + \begin{bmatrix} 1 \\ h_3(\frac{1}{R_1}+\frac{1}{R_2}) \\ h_3 \end{bmatrix} I[N]. \tag{2.34}$$

This system is of third order, evidencing the introduction of one more state by the RC network in the load. It may still be stabilized by using mathematical tools to compute eigenvalues of the A matrix for h_3 and h_4 gain values.

Discrete-time approximation methods are presented in [2] which may aid to incorporate non-linear loads or loads with multiple inputs or outputs into the current control loop.

2.2.4 Control Features

Several additional concerns arise with the PI current controller. These issues are addressed, and the complete PI current control methodology is shown in Figure 2-11.

Excessive Levels

When the output current exceeds the maximum limit as specified to avoid damaging the electronic devices, the controller ceases to operate. The voltage loop supplies the inner current loop with the command $k = 0$. When $k = 0$, then the transistor of the boost converter will never turn on. The boost converter output voltage will be reduced to the amplitude of the rectified input voltage.

Soft Startup

On startup, the error in the output current is high, resulting in large voltage reference commands and undesirable output current overshoot from the PI compensation. In a "soft startup," the output current is increased slowly until a target is reached, alleviating the problem of overcompensation and overshoot. In the prototype, the output voltage is mapped such that only voltages above $278V$ correspond to non-zero A/D input voltages. Therefore,

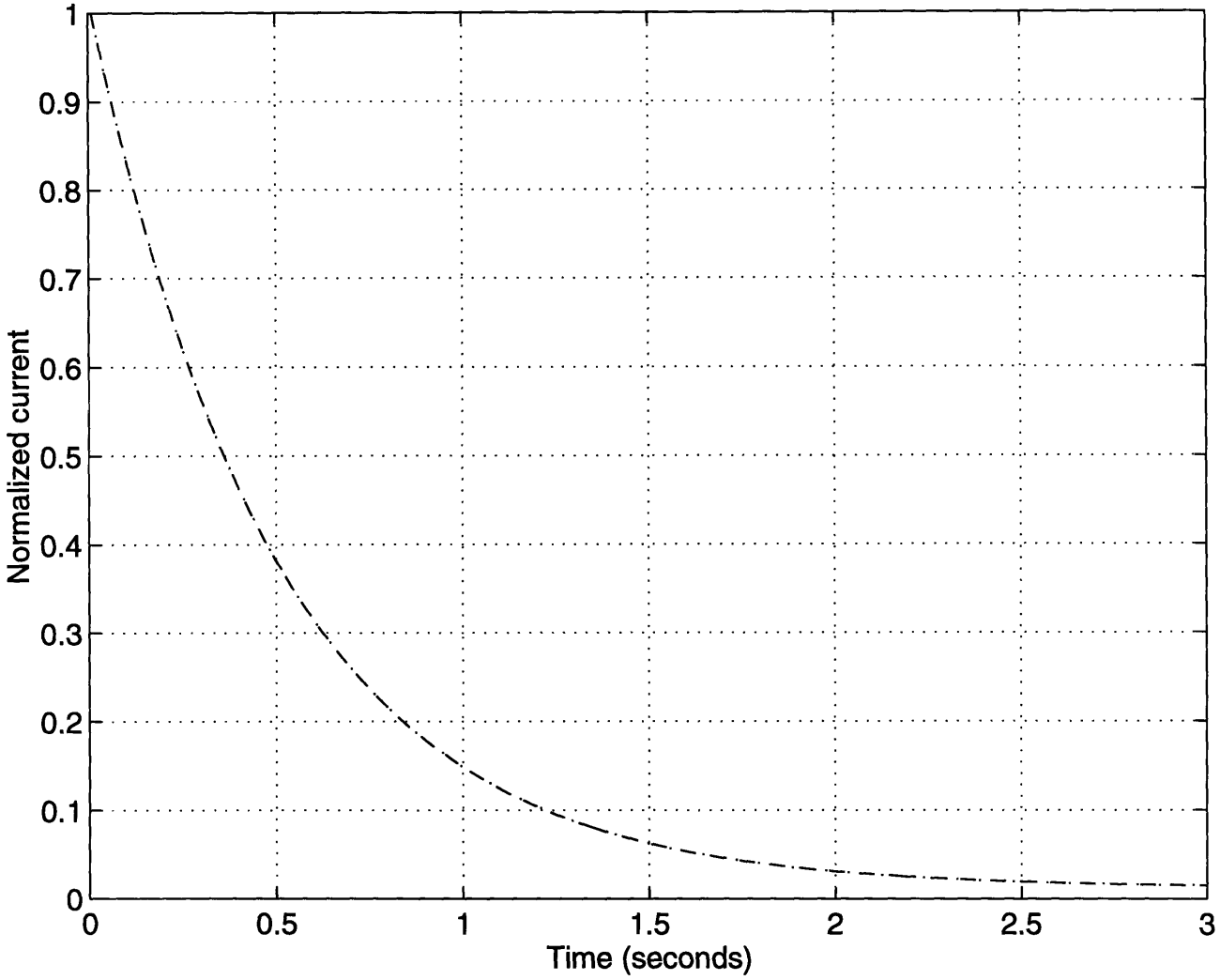

Figure 2-10: Comparison of CT and DT step responses with step-invariant transform.

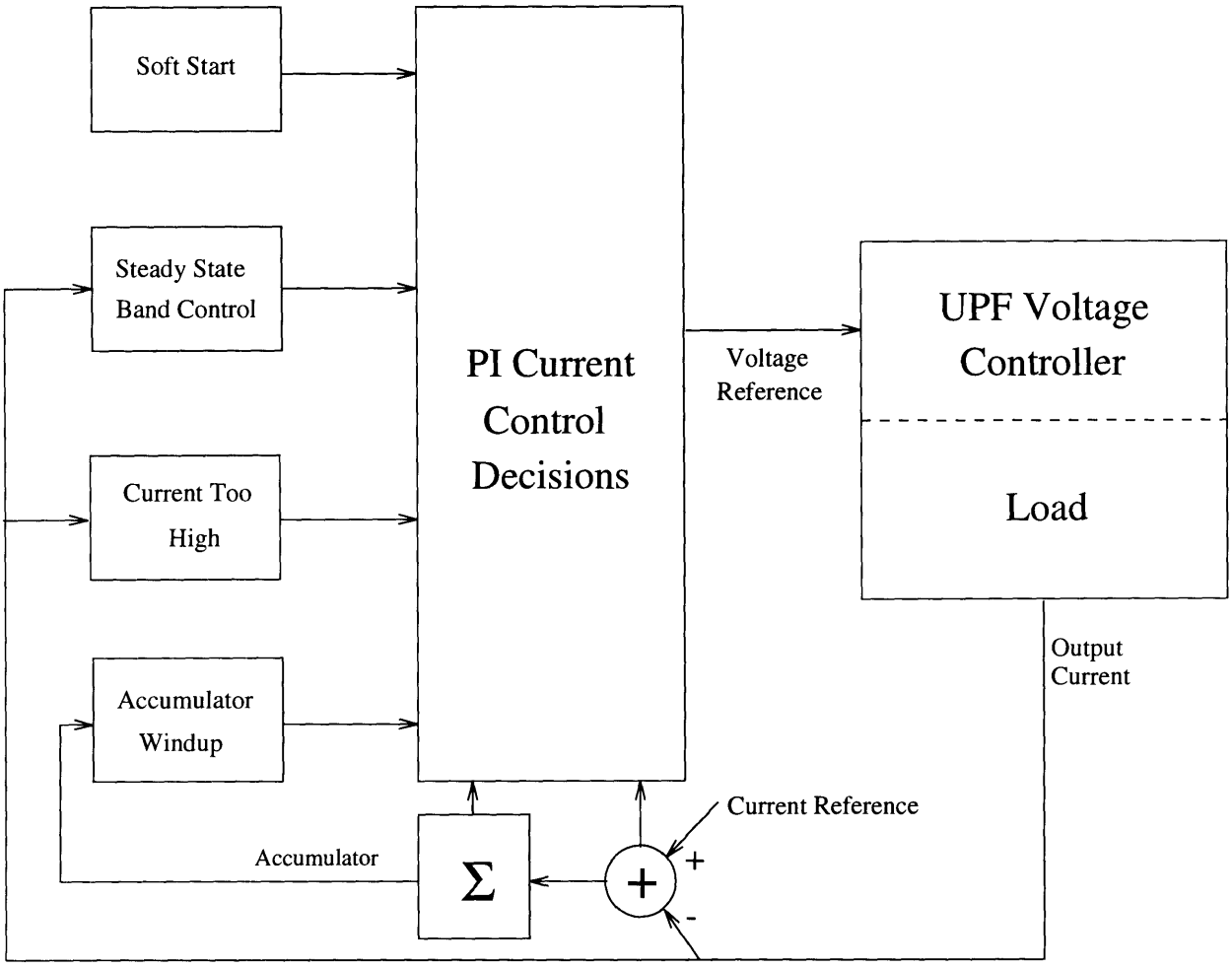

Figure 2-11: Current control with added features.

during startup, while the output *voltage* is less than $278V$, closed loop control is impossible and instead the command k is increased monotonically in open loop. While the output voltage is increasing, so is the output current. When the output voltage reaches $278V$, then the controller can operate in closed loop, and completes the soft start by ramping the output current reference slowly until a target is reached. For the transition from open loop to closed loop to be smooth, a set of predetermined values are loaded into the digital voltage and current accumulators and references. The PI controller will operate as though it had been running in closed loop even during the open loop segment.

Accumulator Windup

Large errors in output voltage and output current will force the controller to compensate heavily by commanding very large or very small input powers. The boost converter system has maximum and minimum input power levels that it can sustain. When the amount of power demanded by the controller exceeds the physical limitations of the UPF supply, then the accumulators will add up large errors quickly while the output voltage and current will approach their targets slowly. Commanding beyond the limits of the system is *saturation*. When finally the PI voltage or current error terms are reversed by exceeding the target (in either direction), the accumulator will have grown excessively large. The accumulators may require an undesirably long time to "wind down" while adding the small negative errors before reaching a final, zero error steady state.

A simple method for avoiding accumulator "windup" is to halt error accumulation while the controller is saturated. The accumulation recommences when the error is small enough for the PI command to be within allowable limits.

Steady State Bands

With a digital implementation, quantization in the PI calculations may result in slightly fluctuating command signals. As a result, the output current may oscillate slightly around a zero error steady state point. This problem is eradicated if an averaged command controls the plant rather than a signal calculated every cycle. As long as the output current remains within a specified steady state band ($\pm 2\%$, for instance), then no control voltage reference is computed. Rather, the voltage reference is taken to be the average of previous control computations. Although this mode of operation runs open loop, closed loop

operation recommences as soon as the output current drifts outside the tight steady state band.

Steady state averaging can occur within the inner voltage loop as well. In this case, the current PI controller must compute the output voltage steady state band values in addition to the output voltage reference.

2.3 Example Gain Calculations

Placing closed loop poles involves solving the gains of the system such that the eigenvalues of the A matrix are the desired poles. Poles with smaller magnitudes correspond to faster transients, and poles with smaller imaginary parts correspond to smaller oscillatory responses.

Voltage Loop

As an example of voltage loop pole placement, let the poles be .95 and .95. From (2.17), the gains are $h_1 = .1$ and $h_2 = .0025$. The eigenvalues (poles) of the state matrix after 50 iterations are $.95^{50} = .0769$.

Current Loop

As far as the current loop is concerned, closed voltage loop poles of .0769 are adequately small for the delay model assumption to hold. For a resistive load $R = 3.91k\Omega$, letting $h_3 = 1958$ and $h_4 = 2150$, from (2.27) the closed current loop poles are .1347 and .3645. These eigenvalues converge quickly, and have been shown experimentally to be reasonable (too ambitious poles will demand more performance from the system than is available).

Chapter 3

Inductive Coupling Interface

Energy from the DC boost converter power supply is inductively coupled through the air gap of a two part transformer, as shown in Figure 1-1. The secondary of the transformer supplies the energy coupled from the primary to the battery load. In this chapter, methods are developed to couple energy through the transformer and to the load efficiently. Design of the transformer and power electronics is discussed.

3.1 DC to AC Conversion

In order to couple energy through a transformer properly, the primary voltage supply must be AC. When a DC voltage v is applied to an inductor with a constitutive law $v = \frac{d\lambda(t)}{dt}$, the flux linkage $\lambda = Li$ will ramp up indefinitely, eventually saturating the core material and causing a huge current to flow [25, p.589-91]. Saturation of the core eliminates the linear relationship between the flux density B and the magnetic field H, $B = \mu H$. Since the energy to be coupled from one part of the transformer to the other is related to the flux Φ, and since $\Phi = BA$ where A is the cross-sectional area, since H increases linearly with current i, and since B increases much less than linearly with H when the core is saturated, then the energy to be coupled increases much less than linearly with i. Thus it is imperative to avoid core saturation both to avoid large currents and to preserve the linear energy to current relationship.

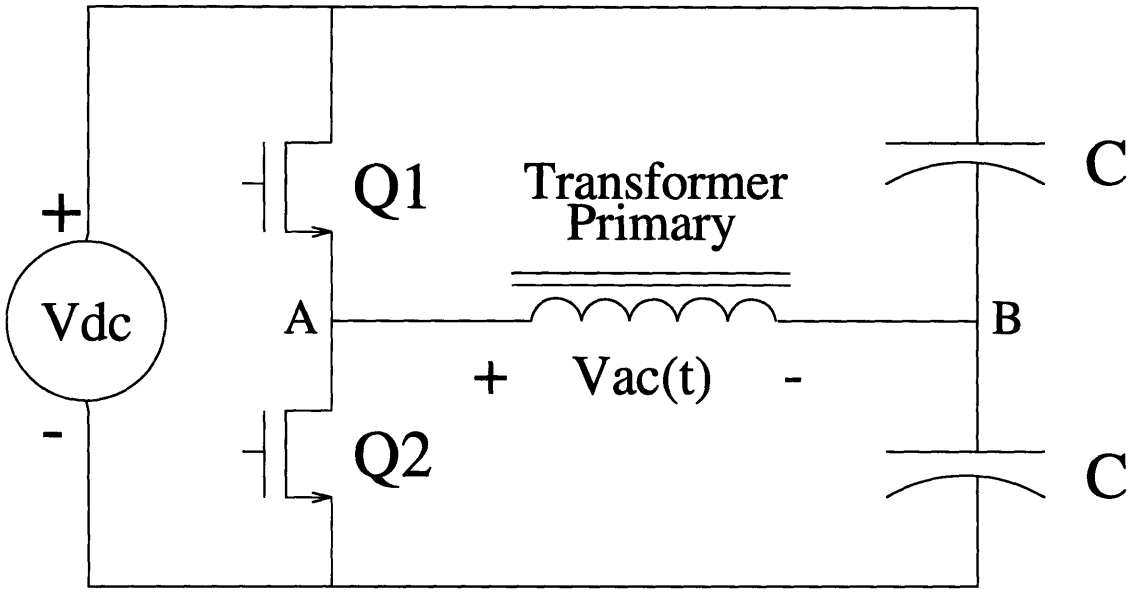

Figure 3-1: Half bridge DC/AC converter. The AC signal is Vac(t).

A half-bridge circuit, shown in Figure 3-1, is employed in the prototype to impress an AC waveform across the transformer primary [35]. The transistors $Q1$ and $Q2$ may be operated so that V_A is connected to ground, connected to the DC bus, or floating. When $Q1$ conducts, there is a short circuit path from V_{DC} to node A. the transistor $Q2$ conducts after $Q1$ has been cut off. At this point node A is shorted to ground. $Q2$ turns off, and the cycle repeats. If the MOSFETs are both conducting, then the catastrophic condition called "shoot through" that V_{DC} will be shorted to ground exists. There must be some small amount of time to allow the previously conducting channel to cut off before the other channel conducts to avoid shoot though.

In the half-bridge circuit of Fig. 3-1, the two large capacitors maintain a constant voltage equal to $V_{DC}/2$ at node B. A full-bridge topology replaces the capacitors with two more power transistor switches. Whereas for the half-bridge case the peak to peak AC voltage swing across the load is V_{DC}, for the full-bridge case the peak to peak AC voltage swing across the load is $2V_{DC}$ with a proper switching scheme. In applications such as frequency converters, proper filtering within the bridge will force V_{AC} to be a sine wave. For this charging application, no such filtering is required as a square wave suffices. Hence no additional inductor or capacitor filter components are necessary.

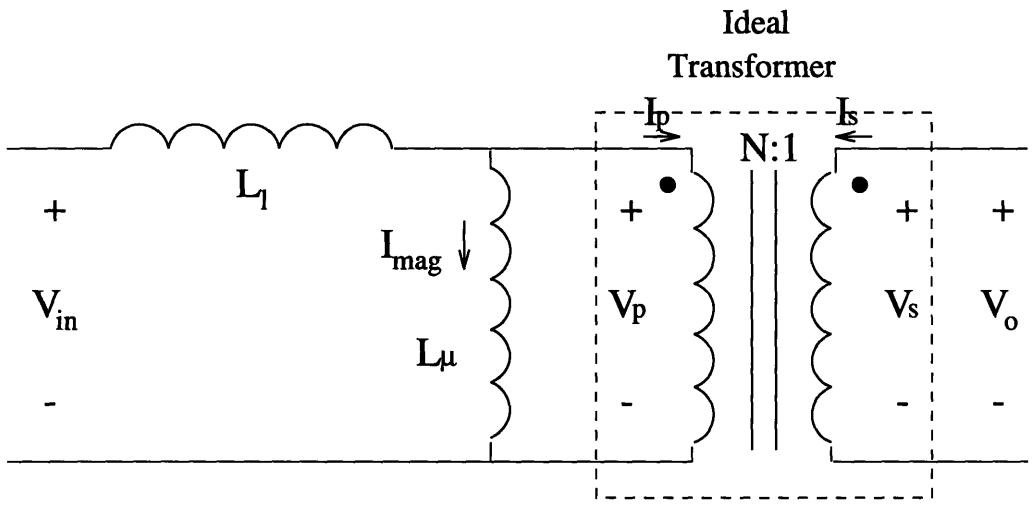

Figure 3-2: Model of a transformer.

3.2 Transformer Model

Understanding the energy recovering switching scheme of the inverter transistors requires an examination of the model of the transformer. Figure 3-2 shows part of a standard transformer model [25, p.591-3]. Within the dotted box is the universal ideal transformer featuring the relationships $V_p/N = V_s$ and $NI_p = I_s$. The magnetizing inductance L_μ is the inductance seen from the terminals of the primary coil that arises from a non-infinite magnetic permeability μ. The coil around the core material forms a solenoid inductor. Calculating this term is possible with knowledge of the transformer geometry. More difficult computational methods using finite element mesh analysis exist for determining the leakage inductance L_l.

While calculations based on knowledge of the geometry of the gapped core transformer lead to theoretical values for L_l and L_μ, an easier method to find those terms is to measure them empirically. A convenient way to find L_l is to measure the inductance at the primary terminal of the transformer with the secondary shorted. Figure 3-3-(a) shows the simplified circuit. The magnetizing inductance can subsequently be measured by subtracting the leakage inductance from the inductance measured at the primary terminal with the secondary opened, as illustrated in Figure 3-3-(b). Unfortunately, the empirical method determines the values *after* the transformer is built. To build a transformer with target values of L_l and L_μ, typically quick, first cut theoretical calculations are carried out *before* construction, and

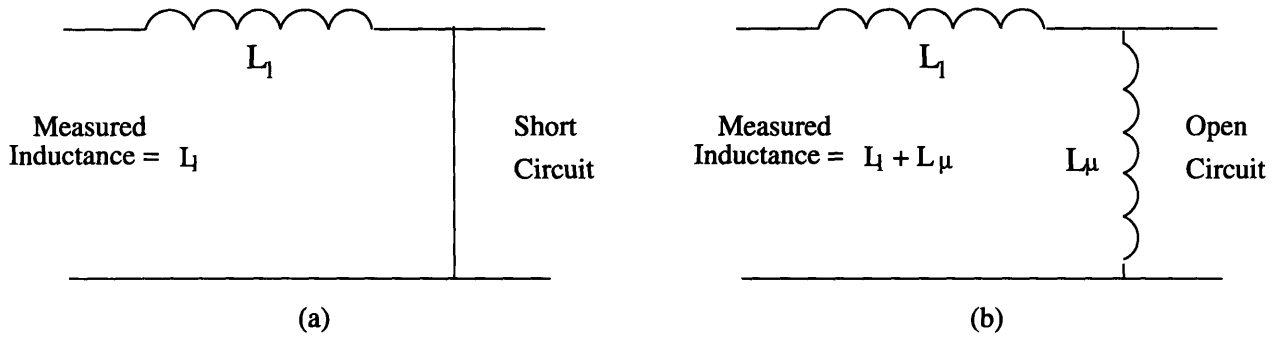

Figure 3-3: Simplifications of transformer model with secondary short circuited (a) and secondary open circuited (b).

empirical measurements are made *after* construction.

With the two impedances $j\omega L_l$ and $j\omega L_\mu$ acting as a voltage divider, a small leakage inductance will not greatly reduce the secondary voltage V_o from the ideal V_{in}/N. However, with the introduction of an air gap in the transformer core, the leakage inductance increases considerably, resulting in smaller secondary voltages. Figure 3-4 schematically demonstrates the undesirable effect of an air gap for a C-core transformer. More flux lines such as those numbered 1, 2, and 3 will avoid the secondary core in the return path to the primary core for a large air gap. The L_l term models the amount of this flux "leakage" [25, p. 593]. The air gapped E-core configuration of Figure 3-5 used in the prototype is equally prone to high leakage inductance effects.

3.3 Secondary Side Rectifier

To supply the battery with a charging current, a full-bridge rectifier as shown in Figure 3-6 rectifies the AC voltage appearing at the terminals of the secondary winding. During the diode conduction period, energy, drawn through inductive coupling from the transformer primary to the secondary, supplies the load and output bus capacitor. When the diodes are off, the capacitor supplies current to the load. The rectifier diodes only conduct when the load voltage droops below the secondary voltage. In the case of a square wave on the secondary, the diodes will always conduct, since the secondary voltage will always remain higher than the output voltage.

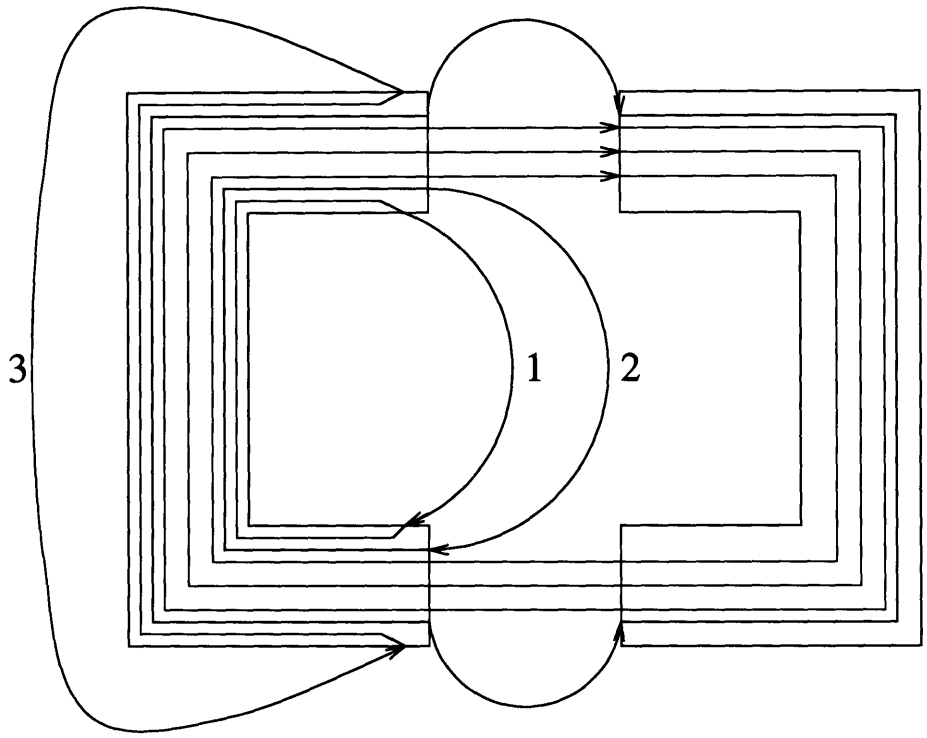

Figure 3-4: Schematic Flux leakage for a C-core. Windings are not shown.

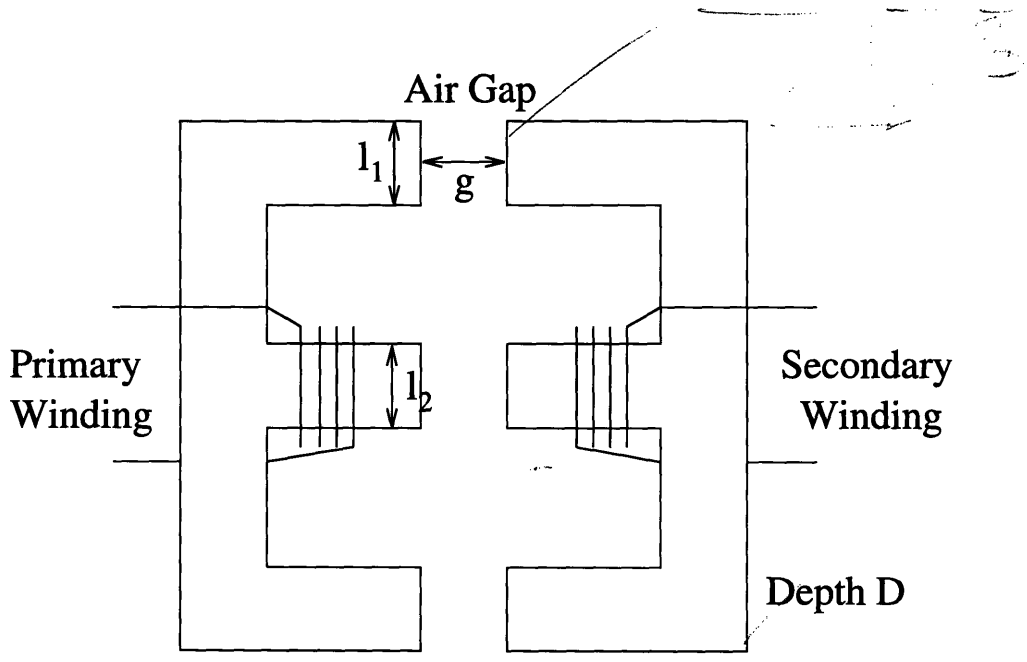

Figure 3-5: Transformer E-core with an air gap.

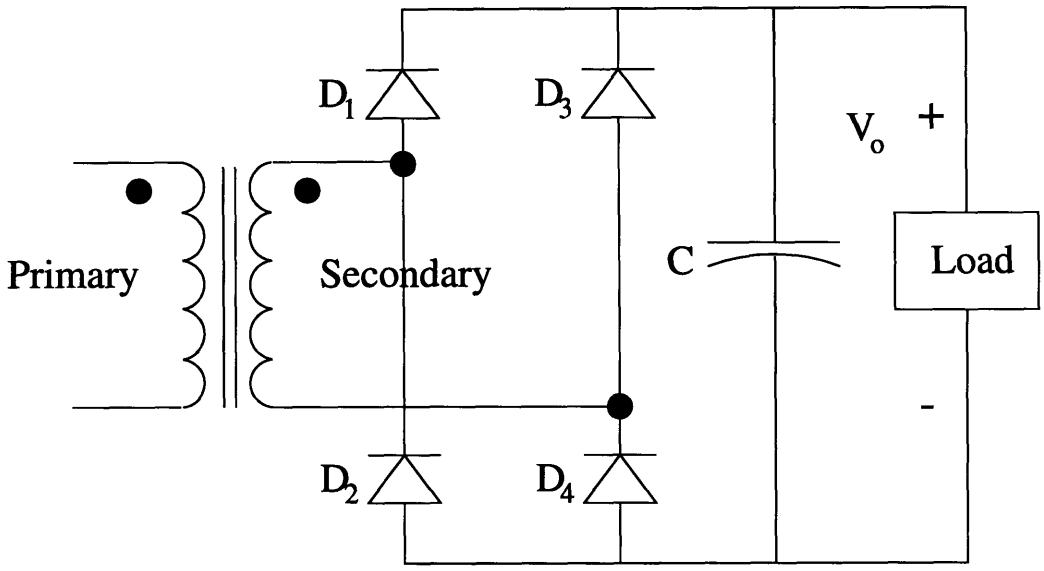

Figure 3-6: Full bridge rectifier on the transformer secondary.

3.4 Transistor Switching Scheme

In typical operation of the inverter and rectifier, the energy stored in the body capacitors of the MOSFETs and rectifier diodes is dissipated through the device channels during switching. Additional losses arise from non-zero voltage switching of the MOSFETs [36]. A switching scheme outlined by Mweene, Otten, and Schlecht [35] utilizes the leakage inductance of the bridge transformer to ensure lossless transitions of the MOSFETs. Power dissipation occurs whenever the VI term across an element is positive. For example, if the voltage across the high side transistor $Q1$ is non-zero at turn on, then there will be a time of positive VI after the channel starts to conduct. Similarly, if the voltage across the transistor increases before the current stops flowing, there will be power dissipation. To reduce switching power loss, the voltage must be reduced to zero before conduction, and must remain at zero until cutoff.

Figure 3-7 shows the connection between the half-bridge inverter and the rectifier. The MOSFET body capacitances are represented by $C_{1,2}$, and diode capacitances are denoted by $C_{D1,D2,D3,D4}$. Typically the energy stored in these capacitors is dissipated in the MOSFET channels and across the diodes when a switch is turned on. The switching scheme from [35] avoids this loss by ensuring the FET channel voltage is zero before turn-on and by recovering the energy in the device body capacitors. The following discussion, which paraphrases the

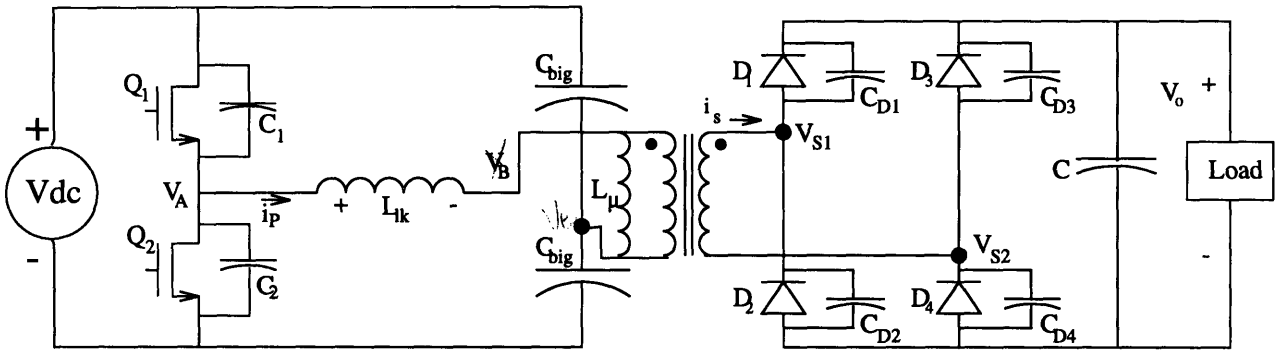

Figure 3-7: Inverter and rectifier with parasitic capacitances.

work by Loveday H. Mweene [37], explains the waveforms shown in Figure 3-8.

Until time t_1, transistor $Q1$ is conducting. The diodes $D1$ and $D4$ are on, and the secondary voltage is V_o. The primary voltage is $\frac{V_{DC}}{2} = \frac{N_P}{N_S}V_S$. The voltage at node B is the sum of $V_{DC}/2$ across the low side bus capacitance and $V_{DC}/2$ across the primary. A small linear slope in i_P arises from the finite magnetizing inductance and $V_{DC}/2$ across the primary. At t_1, $Q1$ is turned off, but the presence of the significant leakage inductance forces the bridge current to continue flowing. A ringing between L_{lk} and the capacitance formed by $C_1||C_2$ results in the discharging of node A. At time t_2, node A is clamped to zero by the anti-parallel body diode of $Q2$ (not shown). Because the diodes $D1$ and $D4$ continue to conduct, V_B still remains at V_{DC}.

During the interval between t_2 and t_4, the primary current decreases linearly since the primary voltage remains $-V_{DC}/2$. Since the voltage at node A is zero, and hence the voltage across the MOSFET is zero, the transistor $Q2$ can at any time be turned on without loss.

When finally the primary current reaches zero at t_4, the diodes $D1$ and $D4$ will stop conducting. The voltage across the entire primary is still $-V_{DC}/2$; the primary current will continue to decline. After t_4 the diode capacitors C_{D2} and C_{D3} will discharge while C_{D1} and C_{D4} charge. A ringing between the reflected leakage inductance and the diode capacitors occurs, at the end of which (t_5) the diodes $D2$ and $D3$ conduct. After t_5, $Q2$, $D2$, and $D3$ are conducting, V_{lk} is zero, and the primary current is reversed and declining slightly.

For $Q2$ to turn on losslessly, the voltage at node A must be able to ring all the way to zero. The energy in L_{lk} at time t_1 must be sufficient for this to occur before the primary

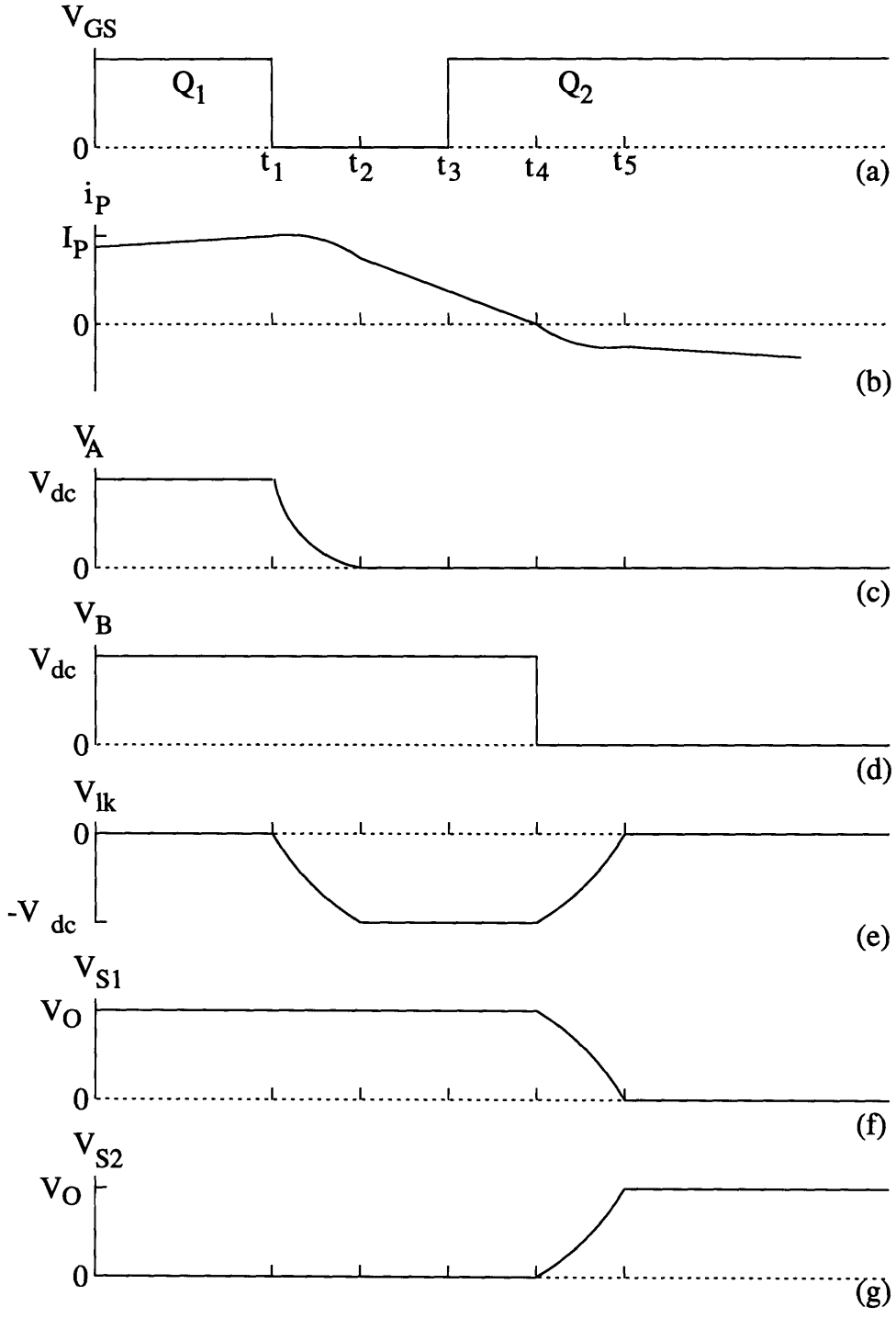

Figure 3-8: Voltage and current waveforms during a switching transition.

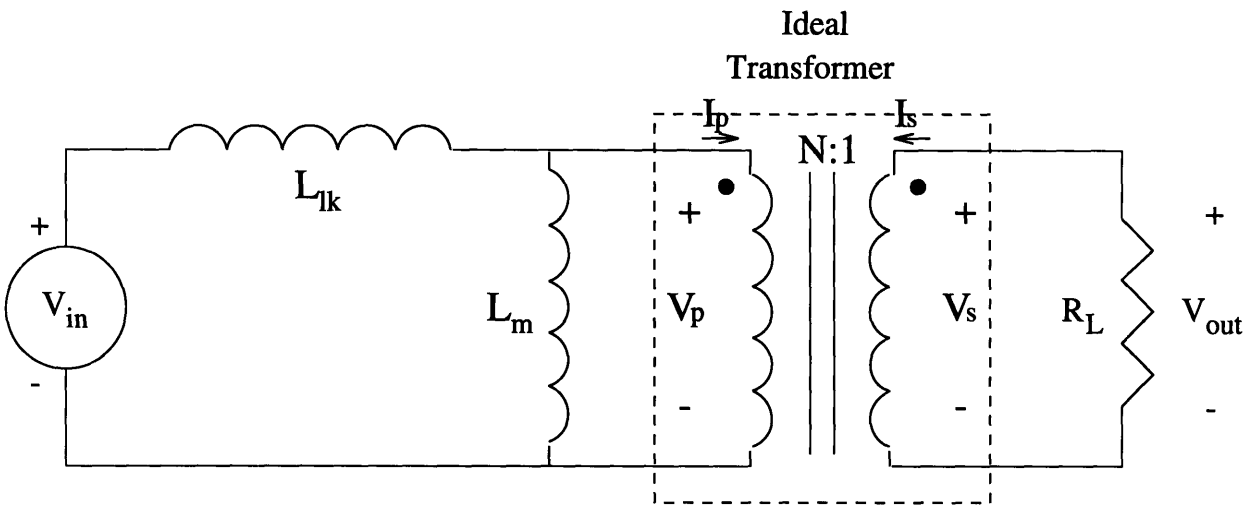

Figure 3-9: Circuit with load resistor and transformer model.

current ramps to zero. Put another way, the leakage inductance must be large enough to ensure that the current drawn from node A when $Q1$ turns off remains non-zero for the length of time it takes for node A to ring to zero Volts. To this end, the leakage inductance can be arbitrarily large.

However, high leakage inductance drastically changes the ability of the transformer to maintain the ideal primary-secondary voltage ratio. For the circuit in Figure 3-9, the transfer function $V_{out}(j\omega)/V_{in}(j\omega)$ is

$$\frac{V_{out}(j\omega)}{V_{in}(j\omega)} = \frac{1}{N}\frac{j\omega N^2 R_L L_m}{-\omega^2 L_m L_{lk} + j\omega N^2 R_L (L_m + L_{lk})}. \tag{3.1}$$

For $N = 250/23$, $L_m = 33.55mH$, $L_{lk} = 5.26mH$, $\omega = 2\pi * 50kHz$, and $V_{in} = 150V$, table 3.1 shows the magnitude of V_{out}/V_{in} for several values of R_L, as well as resulting V_{out} amplitudes and output powers. Clearly, the leakage inductance presents a severe problem for maintaining the same voltage on a wide range of possible loads.

A high leakage inductance creates other difficulties. Figure 3-10 models the behavior of the secondary after the transition has completed [35]. The secondary voltage will differ slightly from the output voltage, thus causing a change over time in the secondary current. If the time constant $\frac{L_{lk}}{R_{eq}}$ is small compared to the switching period, then i_S will follow the dotted line in Fig. 3-10-(b) [35]. For large leakage inductances, i_S will follow the solid line.

R_L	$\lvert V_{out}/V_{in} \rvert$	V_{out} Amplitude	Output Power
∞	.0795	$11.9V$	0
1000Ω	.0795	$11.9V$	$.142W$
100Ω	.0790	$11.8V$	$1.40W$
10Ω	.0507	$7.60V$	$5.78W$
5Ω	.0304	$4.56V$	$4.16W$
2Ω	.0130	$1.95V$	$1.90W$
0	0	0	-

Table 3.1: Effect of leakage inductance for different load resistors.

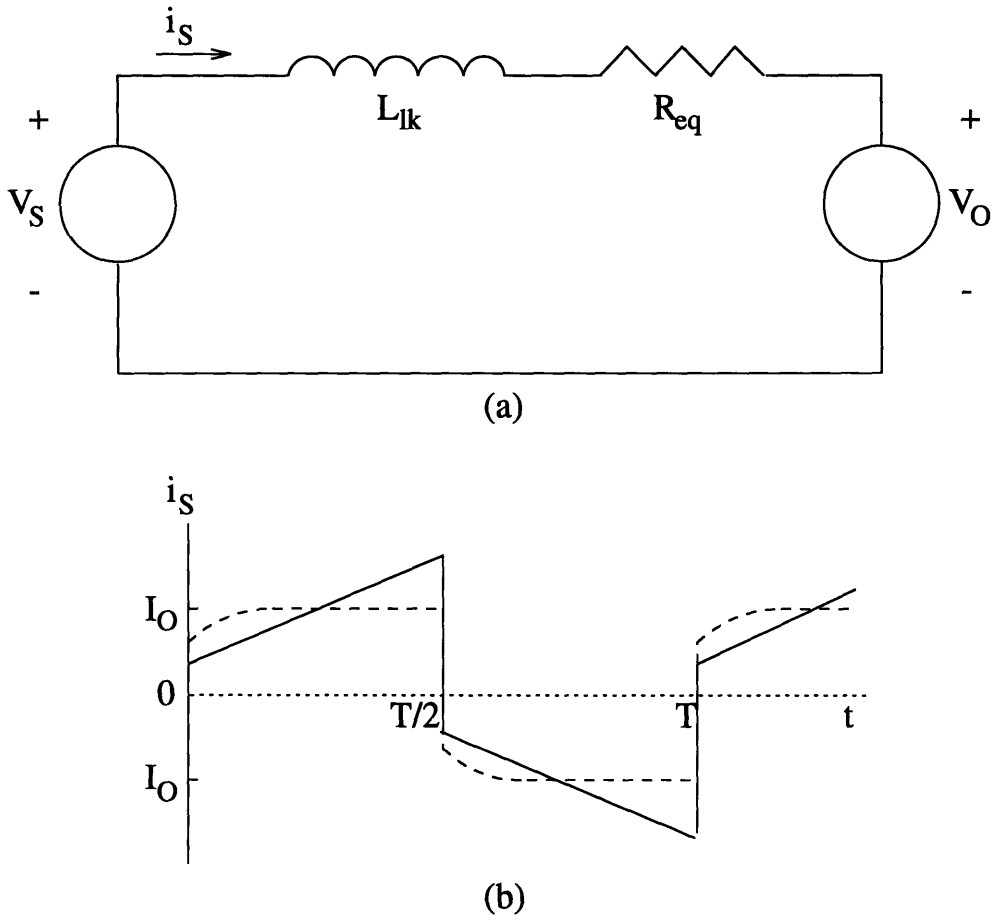

Figure 3-10: Equivalent circuit of converter during conduction.

The starting value for i_S is typically less than the load current, and is smaller for larger values of L_{lk}. The slope of the solid line will be greater for higher leakage inductances, and the final value of i_S will therefore be higher. MOSFET channel losses and other conduction losses in the circuit will grow. Hence there is an upper limitation for practical values of L_{lk}.

3.5 Transformer Design

The size and weight of the transformer are to be minimized since the electric vehicle houses the secondary transformer core. Because the size of a transformer is directly related to how much energy it can store, minimizing the maximum field intensity, which minimizes the maximum core energy, minimizes the necessary size and weight of the transformer. The flux linkage at the primary terminals is the integral of the terminal voltage from the constitutive law $v(t) = \frac{d\lambda(t)}{dt}$). Since the flux density B is linearly related to flux linkage by $B = \lambda/NA$, then minimizing the maximum flux linkage minimizes the maximum magnetic flux density, and thus the maximum energy. By impressing a high frequency voltage square wave at the primary, the amount of time, equal to one half-period, for the flux linkage to ramp up is small. An upper limit for the frequency exists because there must be enough time for the MOSFETs to switch. For the actual transformer and associated power circuitry built for the prototype, the AC square wave was chosen to be 50kHz.

Design of the transformer requires different considerations depending on whether an AC current or voltage source drives the primary. The chief goal is to avoid core saturation, which occurs then B is too high. Transformer efficiency is highest when the B, H relationship is linear (non-saturation).

In general, the flux linkage at a winding can be written as $\lambda = GN^2i$, where G represents geometrical and material parameters, N denotes the number of turns, and i symbolizes the current. The magnetizing inductance is $L = \lambda/i = GN^2$. Specifically, for an ungapped transformer, the flux linkage is

$$\frac{\mu A_e}{\ell_e} N^2 i = \lambda_u = G_u N^2 i \tag{3.2}$$

where μ is the permeability of the core, A_e is the effective area, and ℓ_e is the effective

magnetic path length. For the gapped core transformer of Figure 3-5,

$$\lambda_g = \frac{\mu_o \ell_2 D}{g}\left(\frac{1}{1+\frac{\ell_2}{2\ell_1}}\right)N^2 i = G_g N^2 i \tag{3.3}$$

where μ_o is the permeability of free space. This expression eliminates the dependence on the permeability of the core. A few example calculations reveals that G_g for large enough gaps is significantly less for gapped cores than G_u is for ungapped cores.

For a current source, the flux linkage λ is equal to GN^2i. Since $B = \lambda/NA$, then higher values of λ result in higher values of B. Therefore the method to avoid core saturation is to consider which transformer core type produces the highest value λ for the same current source. Since G is higher for the ungapped core, and since N^2i is the same in both cases, then λ is higher for the ungapped case. Then an ungapped transformer designed to work with a current source without saturating will lead to a conservative design for larger gaps.

In the case of the prototype battery charger in Figure 1-1, the DC / AC converter impresses an AC voltage waveform on the transformer primary. For voltage sources, the maximum value of the flux linkage λ is known from integrating $v(t) = \frac{d\lambda(t)}{dt}$ regardless of whether or not the core is gapped. Thus N^2i for the gapped core is larger than N^2i for the ungapped core by a factor $\frac{G_u}{G_g}$. Designing for a max I_{mag} for the gapped core then automatically enforces a smaller I_{mag} for the ungapped core. Thus a gapped transformer designed to work with a voltage source without saturating will lead to a conservative design for smaller gaps. This is an important point in the field where gaps may vary somewhat.

Chapter 4

Implementation

This chapter describes the details of implementing the digital charging current controller hardware and software, and gapped transformer system hardware.

4.1 Current Loop Controller Hardware

Figure 4-1 summarizes the topology of the three control loops of the unity power factor DC power supply in Figure 1-1. The control loops interact to control the output current of the boost converter and to ensure unity power factor operation. The innermost current loop matches the boost converter input current to a reference, supplied by the surrounding voltage loop, which is a scaled version of the input voltage. The outer PI voltage loop controls the boost converter output voltage to track the voltage reference supplied by the outermost current loop. This section describes the hardware for the boost converter, and each of the three control loops. The interconnection of the hardware systems is shown in

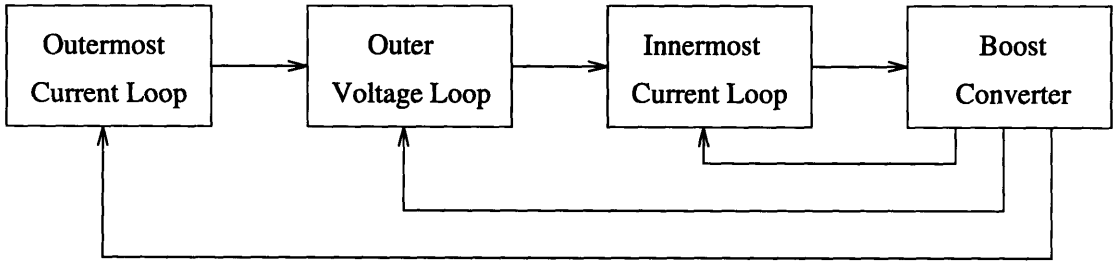

Figure 4-1: Block diagram of the boost converter and the three nested control loops.

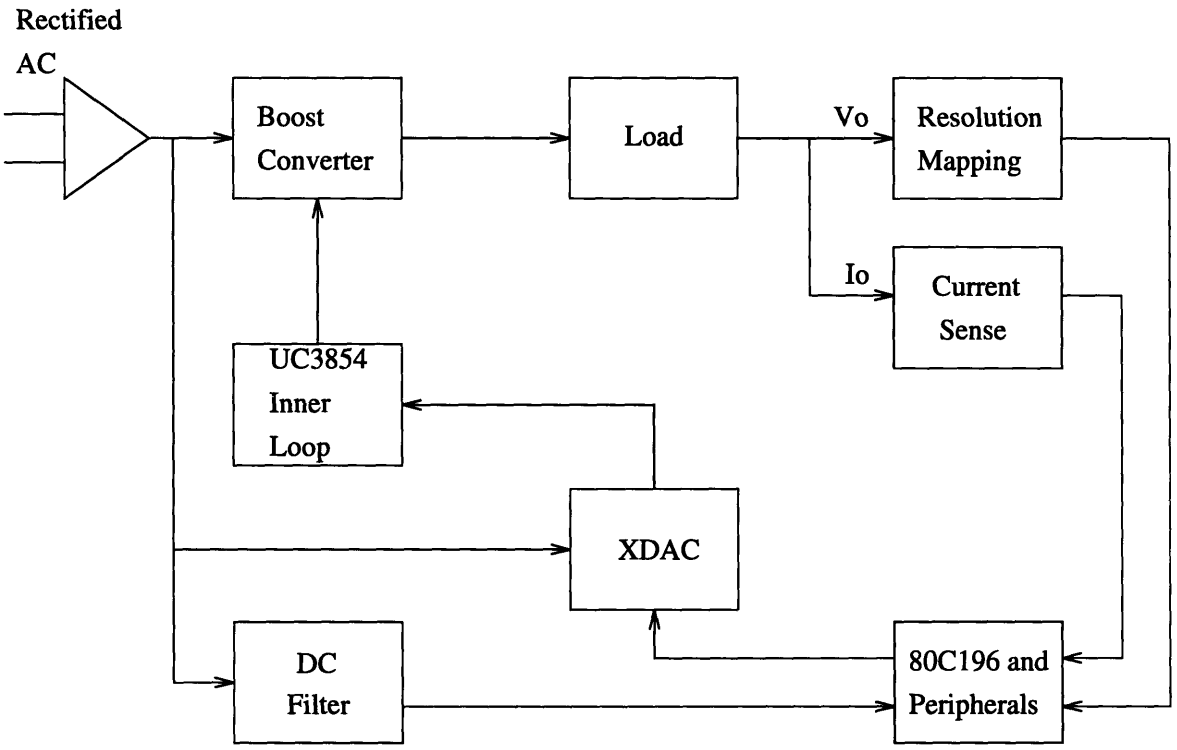

Figure 4-2: Overall structure of hardware system.

Figure 4-2 [34]. The following subsections outline the different parts of figure 4-2.

4.1.1 Accomplishing Unity Power Factor

The inner current loop controller is an analog circuit centering around the Unitrode special purpose IC UC3854 [11] [45]. Figure 4-3 [34] shows part of the Unitrode power factor corrector interface to the system. Three external inputs to the chip, A, B, and C, provide a current reference as the output of the function $\frac{AB}{C}$ to a current-mode controller on the chip. The UC3854 generates a switching pattern for the MOSFET in the boost converter that forces the input current to match $\frac{AB}{C}$. Allowing A and C to be fixed, then controlling B amounts to controlling current proportionally. When a scaled version of the rectified input voltage is fed into input B, then the input current will follow its shape. The Unitrode circuit operates internally at a high frequency, so the slowly varying voltage waveform at $120Hz$ (rectified) can be tracked with very little delay.

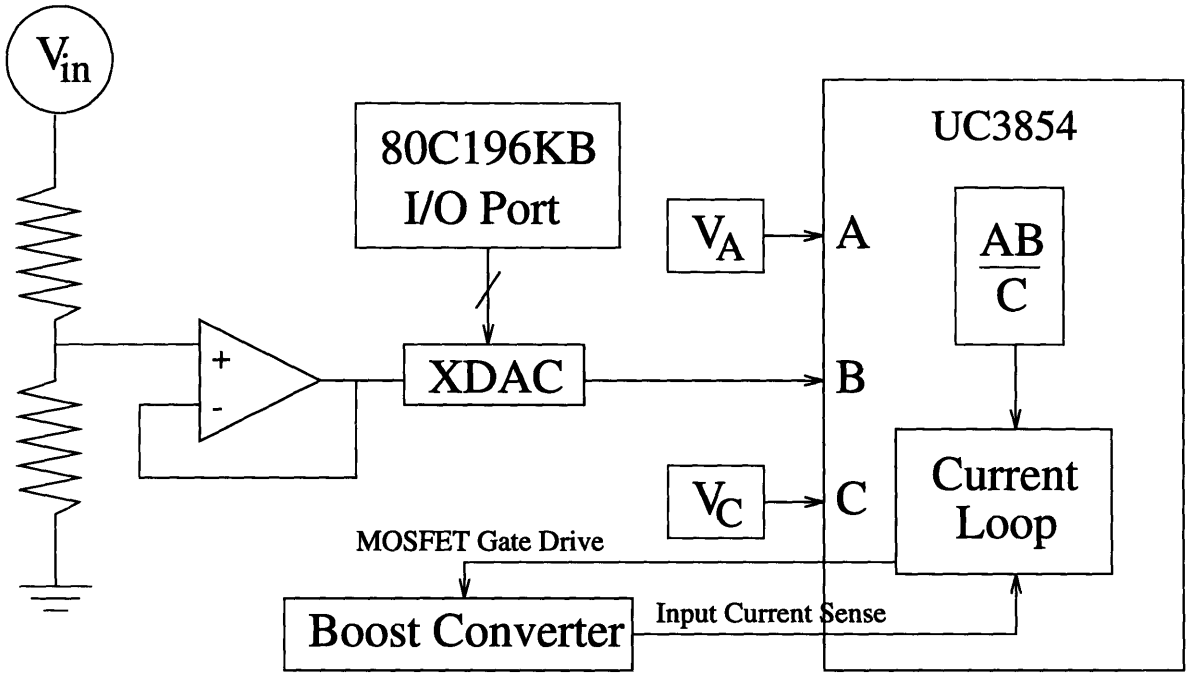

Figure 4-3: Implementation of the inner current loop.

In the prototype, inputs A and C are connected to fixed voltage references created with buffered dividers. The input B has a maximum peak value based on a multiplying DAC. Thus the maximum input current can be adjusted by adjusting the input A. This added feature provides a safety measure, since by setting A there will be a maximum allowable amount of input power drawn from the source. Even with this safety feature, however, care must be taken in testing potentially incorrect or too demanding control algorithms, since too fast *changes* in energy levels may damage elements such as the boost converter capacitor or inductor. Fast changes in the boost converter inductor current, for instance, may result in voltage spikes at the inductor terminals that could damage the MOSFET or diode.

4.1.2 Boost Converter

The boost converter contains a $470\mu F$ capacitor rated to $450V$. The diode and MOSFET switch are the Motorola MUR1560 and International Rectifier IRFP450. A standard packaged full wave rectifier connects the utility input to the inductor. Operating a boost converter in continuous conduction with a "square wave" of duty cycle D (transistor is on a fraction D of the period) results in the relationship $V_o = \frac{1}{1-D} V_{in}$ [25, p. 116]. In this application, the

transistor switching pattern is generated by the modified Unitrode UC3854 rather than by a simple adjustable duty cycle square wave timer circuit.

The hand wound inductor measures $1.025mH$. It has 107 turns with two Micrometals T225-8/90 toroidal cores. The specification claims that this material provides $42.5nH/N^2$ for one core [32]. For $N = 107$, the expected inductance is $.487mH$ for one core, and $.973mH$ for two cores. Alternatively, the specification lists a relative permeability of about 35 at $10kHz$, a magnetic path length of $14.6cm$, and cross-sectional area of $1.42cm^2$. According to the formula for the inductance of a toroid,

$$L = \frac{\mu N^2 A}{\ell}, \tag{4.1}$$

the inductance, with twice the cross-sectional area, is $.980mH$. Both estimates from the data sheet are within 5% of the measured value.

4.1.3 Digital Implementation

Whereas the inner current loop control resides completely in analog hardware, the PI control calculations in the voltage and outer current loops occur within a digital computer. The discrete representation of the state equations leads naturally to a digital implementation of the control calculations. This section describes the capabilities of the digital computer that performs the PI control in the prototype.

Embedded Microprocessor System Overview

The 16 bit Intel 80C196KB register-to-register embedded applications microprocessor evaluation board was chosen [21]. While the processor does not feature the signal processing capabilities of some DSP chips, it is less expensive and provides more than adequate processing power for the control desired in this application. The many faceted peripheral network supports interrupt processing, A/D conversion, timing, and I/O. These facilities are accessed through memory mapped special function registers (SFR). An asynchronous communications protocol between the embedded processor board and a host PC allows easy downloading and executing of code and adequate monitoring of data. The EV80C196KB software package includes the Embedded Controller Monitor (ECM) which supports debug-

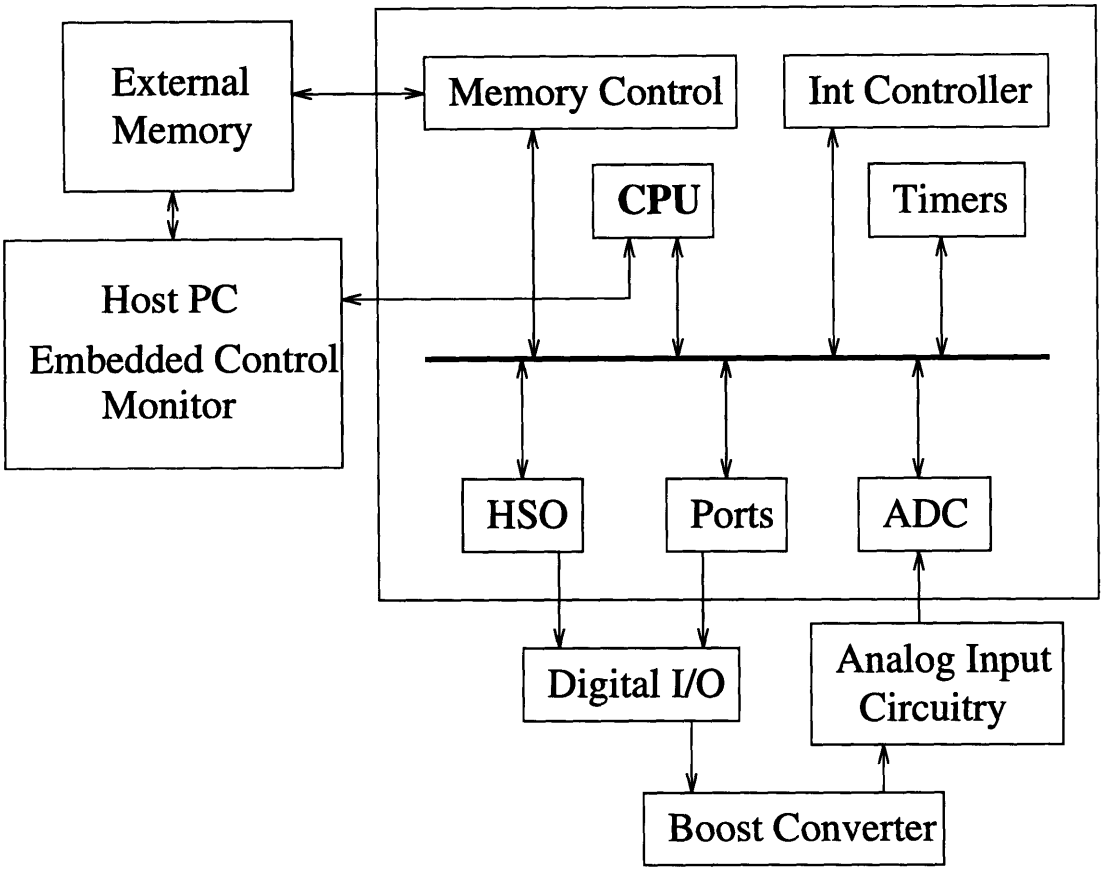

Figure 4-4: The digital development system.

ging facilities for the host computer [20]. The evaluation board with its relevant features is shown in Figure 4-4 [34].

Interrupt Handling

When events such as output voltage A/D conversion completion happen, immediate processing often must be done. When these events are not happening, other, less impending computations can occur. An event that requires attention is an interrupt, and the subsequent processing is the interrupt service routine (ISR). In some applications the ISRs perform minimal amounts of work since the background code requires heavy processor utilization. Other applications such as this one contain ISRs that perform most, if not all of the important computations. In this application, A/D conversion and timing are necessary. As soon an A/D conversion is completed, an interrupt is generated, letting the processor know that the digital word can be read and used. Certain other events such as the commencement of an A/D conversion must occur at exact time intervals apart from each other. Timer events

trigger interrupts which instigate these events.

A/D Conversion

The 80C196KB evaluation board provides 8 multiplexed A/D channels for sampling 8 analog 0-5 Volt signals at different times. For the $16MHz$ processor board, a 10 bit conversion takes approximately $26\mu s$ [34]. To generate an A/D conversion, the SFR AD_COMMAND is loaded with the channel number and the GO bit is set. After the conversion is completed, AD_RESULT_HI contains the 8 most significant bits, and AD_RESULT_LO contains the 2 least significant bits. AD_RESULT_LO also reports which channel's conversion occurred. In this application, three A/D channels are used to sample a scaled version of the peak value of the input voltage, a scaled, resolution enhanced version of the output voltage, and a voltage representing the output load current of the boost converter.

Timers

There are two timers on the 80C196KB. Timer1 is incremented every 8 processor states (16 clock cycles). Its resolution is $1.33\mu s$. Timer2 connects to an input pin for external clocking. With the high speed output (HSO) unit, four "software timers" are available. By programming an interrupt to occur after a certain number of Timer1 ticks, exact timing of certain events can occur. Timer interrupts are prepared by first loading the SFR HSO_COMMAND with the number of the software timer to be initiated, and then writing the value to the SFR HSO_TIME at which Timer1 should trigger that software timer. The triggering of that software timer generates the timer interrupt.

Output Ports

There are four output ports on the 80C196KB. Ports 3 and 4 are outputs used for memory access, while Ports 1 and 2 can be configured as software usable 8 bit outputs. While, as will be discussed, the output in this application requires 12 bits, only a single 8 bit port is necessary since the 12 bit write is done in two steps. HSO output pins are also used for control signals to a multiplying DAC.

Interfacing the Processor and the Boost Converter

As always with digital control of analog hardware, there is an interface which enables computer computation to affect the operation of the circuitry. In this application, the innermost current loop analog controller in Figure 1-3 forces the input current to follow the input

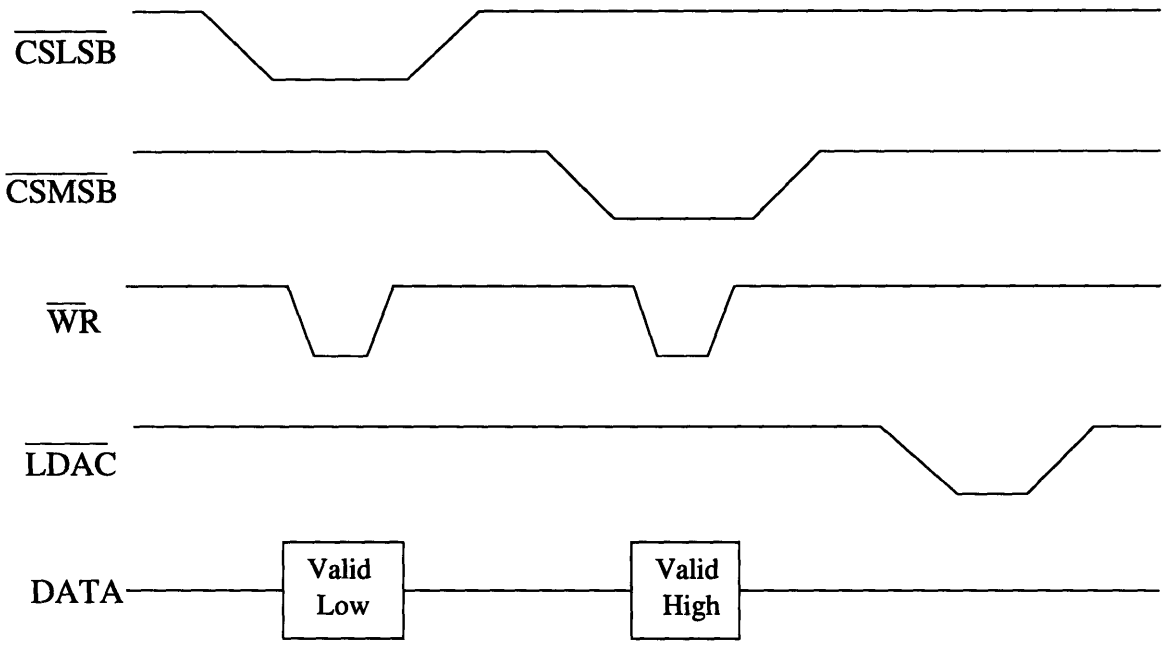

Figure 4-5: Timing diagram for XDAC control signals.

voltage based on a reference provided by the voltage loop. This reference is the rectified input voltage scaled by some analog circuitry and a MAX501 multiplying digital-to-analog converter [30] (input B in Fig. 4-3). The output of the multiplying DAC is $(\frac{n}{4095})v_{in}$, where n is the 12 bit value contained in the DAC as written from Port 1 on the 80C196KB evaluation board. The control bits are written to the DAC sequentially directly from the HSO output pins, as shown in Figure 4-5.

4.1.4 Resolution Enhancement

One of the major causes of quantization is limited numbers of bits in the A/D and D/A conversions. Recognizing that there is a limited range of useful boost converter output voltages that will be impressed upon the terminals of a functioning battery, then mapping this limited range of output voltages to A/D convertible values from 0 to 5 Volts will increase the bit resolution [34]. Figure 4-6 shows such a mapping. First the boost converter output voltage is scaled by a resistor divider to lie between 0 and 5 V. When there is no resolution mapping, then the slope m is $m_o = \frac{5}{3.90} = 1.28$. With mapping, the slope is $m = \frac{5}{3.90 - V_c} = 4.20$, where V_c in this case is $2.71V$. The bit resolution enhancement in number of bits added is $\log_2(\frac{m}{m_o})$,

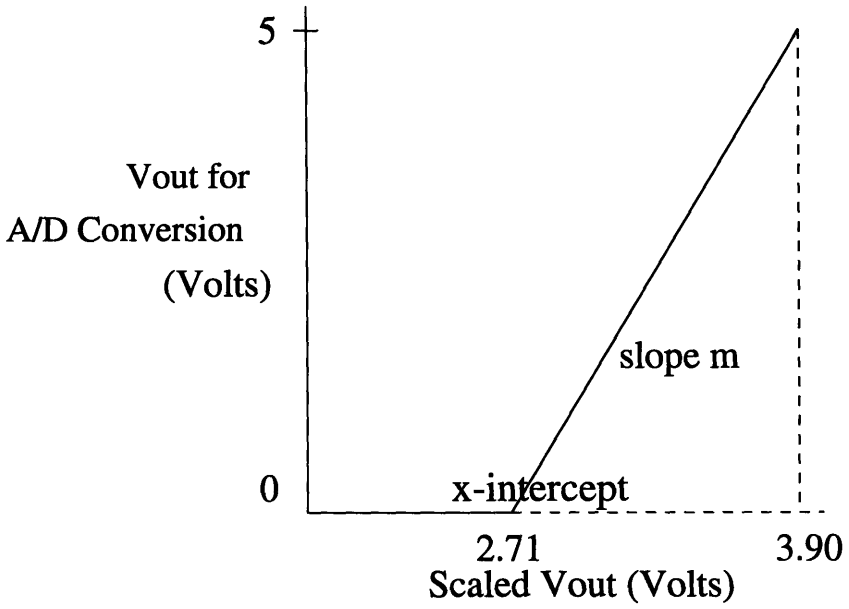

Figure 4-6: Increasing output voltage resolution for sampling.

which in this case is 1.7 bits added. It is arguable as to whether using a 12 bit DAC is useful since there are only 10 significant bits of A/D input.

Scaled outputs smaller than 2.71 Volts are unrecognized by the sampling. When the sample reads zero, then an open loop control state must be entered, since no information is available about the output. Care must also be taken to ensure that the voltage not rise above 450 Volts, because an analog voltage greater than 5 Volts can damage the A/D channel on the 80C196KB.

4.1.5 Analog Filtering

To prevent aliasing from sampling the output voltage at $120Hz$, a low pass filter is connected to the scaled boost converter output voltage. This filter eradicates any $120Hz$ ripple in the output voltage, and eliminates noise generated by power supply spikes, RF, and other high frequency agents. The filter is a sequence of three voltage follower buffered RC-RC stages as displayed in Figure 4-7. Appendix B discusses the design of this filter.

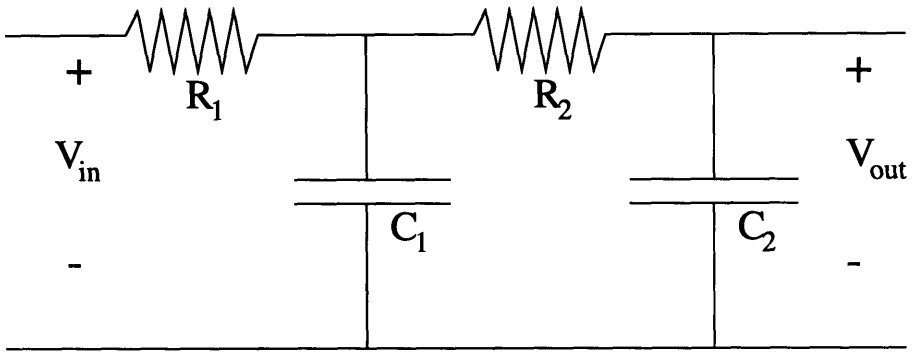

Figure 4-7: Low pass filter stage.

4.1.6 Current Sensing

An LEM current sensing module is used to develop a voltage that is linearly proportional to the boost converter output current. This device calculates the output current by sensing the magnetic field created by the current. The current sensor sends a scaled version of the current through a resistor. The output of the sensing unit is the voltage developed across the resistor. While the LEM module is designed for measuring currents of up to 50 Amps, it can be used for the small amperage for the prototype. To account for offset problems, an op amp adder circuit scales the sum of a buffered adjustable voltage and the output of the LEM module. The final voltage is conditioned to lie within the A/D convertible 0 to 5 Volt range. A low pass filter similar to the output voltage filter is used to prevent aliasing, and is described in Appendix B. Figure 4-8 shows the current sensing system.

4.1.7 Providing Power

Three DC power supplies are used to provide power to the hardware components of the charging current controller. A $+15, -15V$ supply powers the analog circuitry, such as the multiplying DAC and filtering op amps. The microprocessor receives power from a $+5, +12, -12V$ supply. Both of these power supplies are standard units. The third power supply was constructed to provide $18V$ to the UC3854. It contains a step down wall transformer, rectifier, bus capacitor, and variable voltage regulator. The use of three power supplies enabled separate testing and activation of the hardware pieces. For example, since the fixed A and C inputs to the UC3854 needed to be present before the power factor corrector could safely

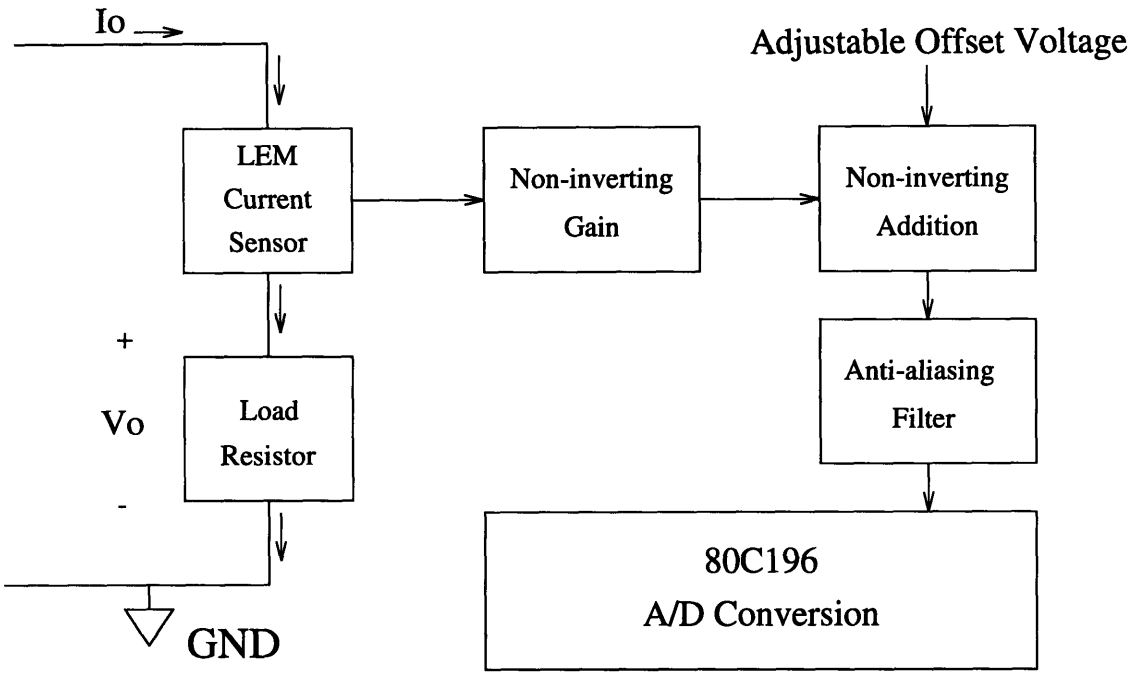

Figure 4-8: Current sensing system.

begin operation, the $15, -15V$ supply was always activated before the $18V$ supply.

4.1.8 Testing Setup

The AC utility input was fed through a VARIAC (variable AC transformer) to allow for adjustment of the input voltage [34]. Figure 4-9 shows the test stand that was used for testing the prototype. AD204 isolation amplifiers protect the fragile digital hardware from overloads and voltage spikes in the analog parts [1]. At the boost converter, a power resistor of $3.91k\Omega$ is the load for the prototype, and another similar resistor can be switched in parallel with the first through a MOSFET. The A input of the UC3854 is adjusted to allow a maximum of 50 Watts input. For $3.91k\Omega$, 50 Watts converts to $442V$. Care was taken in the control code to ensure that the output voltage not exceed $450V$, the boost converter capacitor rating. Detailed circuit diagrams can be found in the appendices.

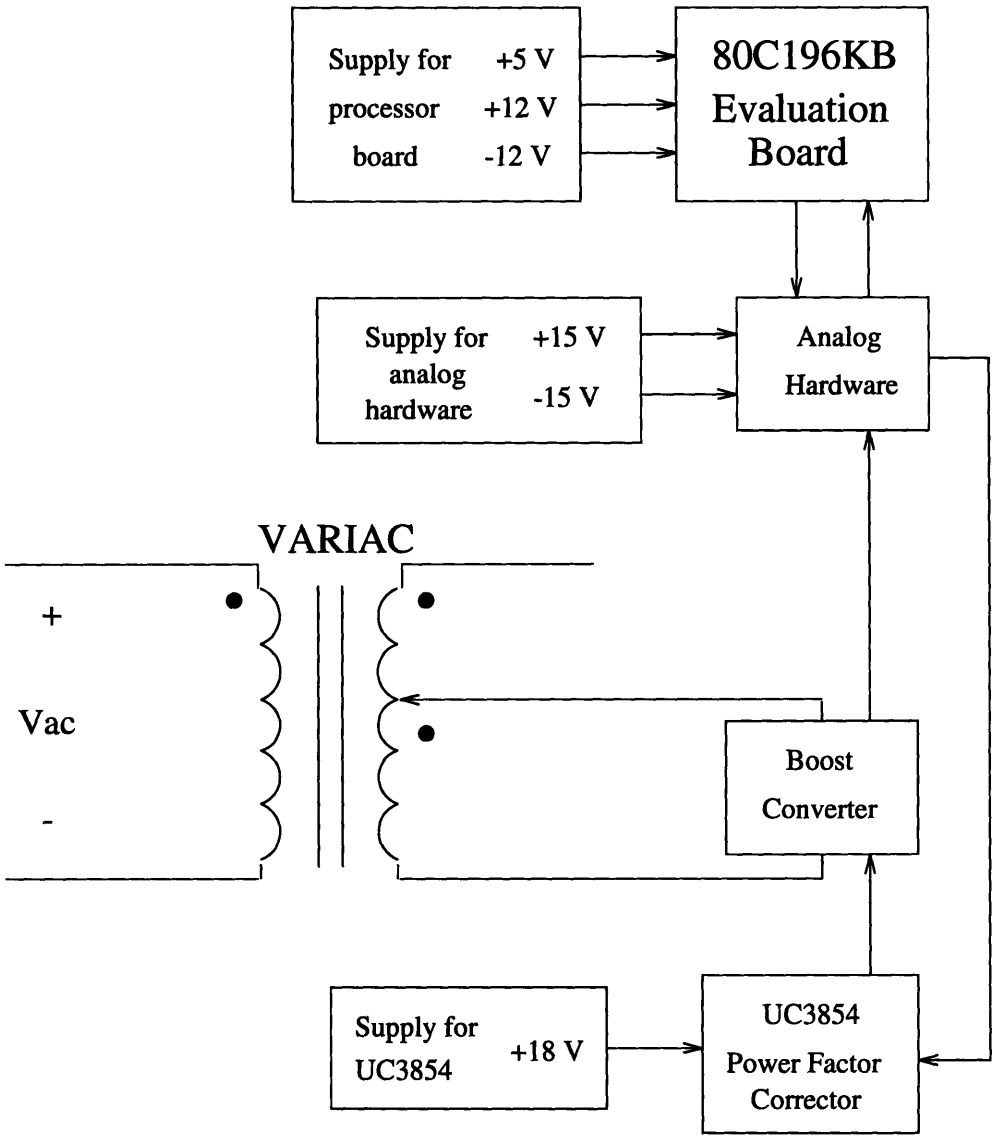

Figure 4-9: Diagram of testing setup with three power supplies and utility .

4.2 Current Loop Controller Software

The software that runs on the 80C196KB embedded processor board accomplishes PI control of the voltage and current loops. This section describes the interrupt service routine structure, scaling of parameters, and implementation of other controller features in software.

4.2.1 Interrupt Driven Routines

The software control of the battery charger revolves around four interrupt service routines. All processing is done in these routines, which are triggered by A/D conversion completion interrupts and a timer interrupt. The following discussion details the activity in each ISR, and is summarized in the flow chart in Figure 4-10.

Timer ISR

For the unity power factor assumption to hold, the command k must be computed once every $1/120 = 8.3ms$. The samples of the output voltage, output current, and rectified input voltage must occur at regular intervals of $8.3ms$. The Timer ISR will ensure proper scheduling of A/D conversions by executing at a frequency of $120Hz$. To maintain this sampling rate, the first instruction of the timer interrupt service routine is immediately to schedule the next timer. Experiments showed that for a period of $8.3ms$, the software timer should count 8406 ticks before the next interrupt is triggered. After scheduling the next timer interrupt, the Timer ISR updates the current reference based on a stored charging current profile, and starts the A/D conversion of the output current.

Output Current Conversion Complete ISR

The output current A/D conversion complete interrupt triggers this ISR. After ensuring the output current is not too high, this routine performs the current steady state band checking. PI control occurs only once every 50 interrupt instances because the current loop operates at a sampled data rate fifty times slower than the voltage loop. The voltage reference and the voltage steady state bands are updated. Finally, the output voltage conversion is begun.

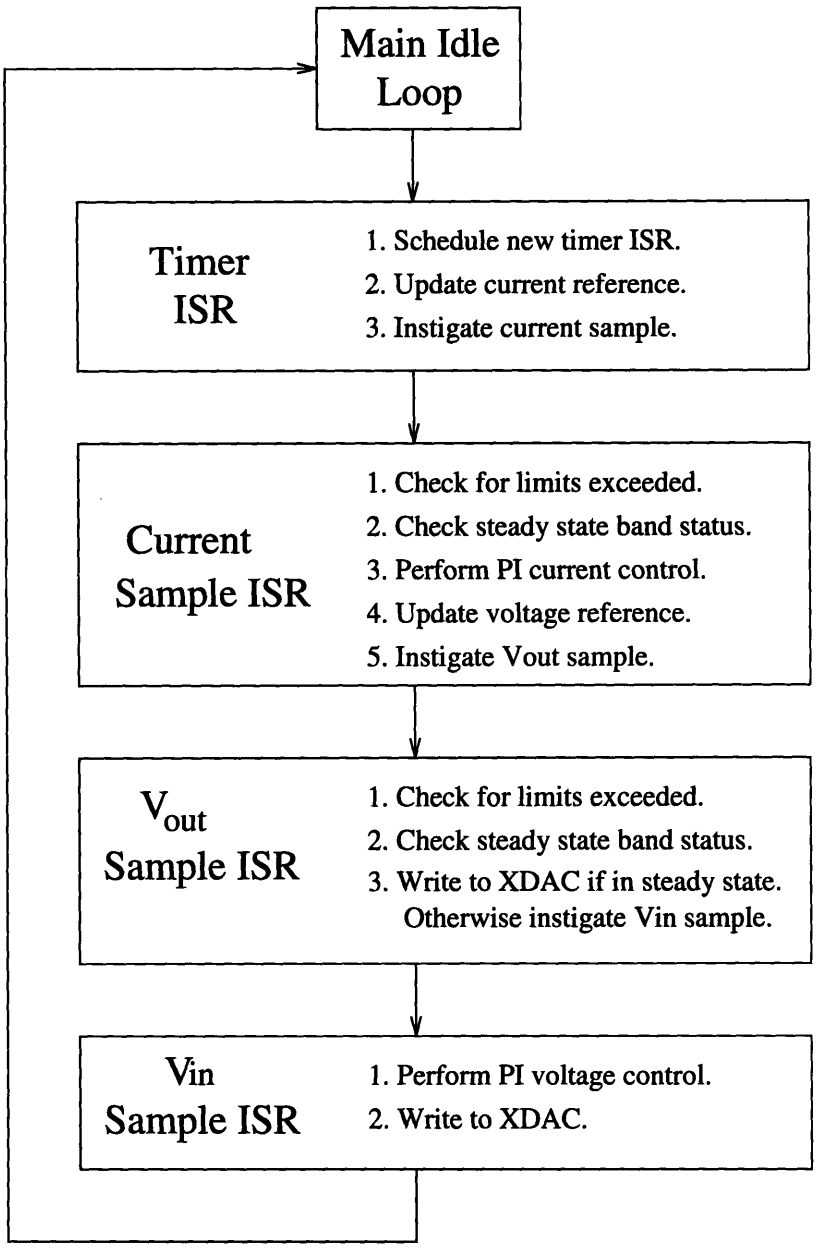

Figure 4-10: Structure of software run-time execution.

Output Voltage Conversion Complete ISR

This ISR, triggered upon output voltage A/D conversion complete, first checks for too high output voltage. The value read in is corrected for effects from resolution mapping. Soft start is effected if appropriate, and steady state band checking is done. If the steady state band condition holds, then the average of past control values is written to the multiplying DAC. Otherwise, a conversion of the input voltage peak value is started. The possibility that the output voltage falls below the threshold for the resolution mapping exists. This ISR will increase k open loop whenever the output voltage drops below the threshold.

Input Voltage Peak Conversion Complete ISR

PI control occurs immediately following the input voltage peak A/D conversion complete interrupt. Anti-windup measures are taken by ceasing the accumulation process if the controller computes a k higher than the maximum, 4095 (from 12 bit multiplying DAC). The resulting value $min(4095, k)$ is written to the multiplying DAC.

4.2.2 Scaling of Parameters

The discrete time equation for the control signal $k[n]$, as found in Section 2.1.5, is

$$\tilde{k}[n] = \frac{C}{T_L V^2}(h_1(X - x[n]) + h_2\sigma_v[n]) + \frac{2}{V^2}P[n]. \tag{4.2}$$

Similarly, the DT equation for the control signal $V_o[N]$, from Section 2.2.2, is

$$V_o[N] = h_3(I_{ref} - i[N]) + h_4\sigma_i[N] \tag{4.3}$$

To implement this compensation scheme on a digital controller, the discrete time equations must be scaled appropriately for the microprocessor embedded within the system to use them. While the PI calculations within the processor are identical arithmetically to the discrete time equations (4.2) and (4.3), the gains and values representing voltages and currents are different. This section explains how to derive the digital gains that provide a representation of the analog signals within the digital controller.

The following list of parameters employs the notation of [34].

- v_o is the boost converter output voltage.
- i_o is the boost converter output current.
- V is the peak rectified input voltage.
- v_{od} is the digital representation of the output voltage.
- i_{od} is the digital representation of the output current.
- V_{id} is the digital representation of the peak input voltage.
- V_o is the reference voltage calculated by the current control.
- I_o is the charging current reference.
- V_{od} is the digital representation of the reference voltage.
- I_{od} is the digital representation of the reference current.
- σ_{vd} is the digital accumulator for the voltage loop control.
- σ_{id} is the digital accumulator for the current loop control.
- h_{1d}, h_{2d} are the digital gains for the voltage loop control software.
- h_{3d}, h_{4d} are the digital gains for the current loop control software.
- h_{pd} is the digital gain for the power term.
- G_{dig} is a digital gain used to account for integer arithmetic in calculating the voltage loop digital gains.
- k_d is the digital current command reference calculated by the voltage control.
- k_{max} is the maximum value k for the system with input current $i_{in} = kv_{in}$. The maximum digital k is 4095, in accordance with the 12 bit multiplying DAC.
- D_{ivo} is the scaling factor for the boost converter output voltage. The input to the resolution mapping circuit is an analog signal which is D_{ivo} times the real output voltage.

- D_{ivi} is the similar scaling factor for the peak rectified input voltage.
- D_{cur} is the scaling factor resulting from the current sensing circuit.
- F_{AD} is the A/D sampling gain. For the 10 bit A/D, the gain is 1023/5. The actual digital number resulting from the conversion is an integer.
- m is the slope of the linear region in the resolution mapping.
- x_{int} is the x-intercept where the resolution enhancement begins to map to non-zero voltages.

Controller Inputs

The three inputs to the controller v_o, i_o, and V are sampled, and require scaling for proper digital representation. Specifically, the output voltage is fed through a resistor divider and the resolution mapping. The equation for the resolution mapping is

$$v = m * (D_{ivo} v_o - x_{int}). \tag{4.4}$$

where $D_{ivo} v_o$ is the divided boost converter output voltage. The result of an A/D conversion is a digital number that is F_{AD} times the analog voltage value to be sampled. Specifically, the value in the computer after sampling the resolution mapped boost converter output voltage is $F_{AD} v$. Within the software it is possible to "unmap" directly to the scaled output voltage by adding $F_{AD} m x_{int}$. The sampled voltage will then be simply

$$v_{od} = m\, F_{AD}\, D_{ivo}\, v_o. \tag{4.5}$$

The second controller input, V, is scaled by a low pass filter whose gain is D_{ivi}. After sampling, the digital peak rectified input voltage is

$$V_{id} = F_{AD}\, D_{ivi}\, V. \tag{4.6}$$

The equations involved with closed loop voltage control contain the squares of the input and output voltages. The digital representation for these parameters are

$$v_{od}^2 = m^2 F_{AD}^2 D_{ivo}^2 v_o^2 \tag{4.7}$$

and

$$V_{id}^2 = F_{AD}^2 D_{ivi}^2 V^2. \tag{4.8}$$

Since the reference voltage computed by the current PI controller, and the voltage loop accumulator are of the same units ($Volts$) as the output voltage, then the digital squared voltage reference can be expressed as

$$V_{od}^2 = m^2 F_{AD}^2 D_{ivo}^2 V_o^2, \tag{4.9}$$

and the digital accumulator as

$$\sigma_{vd} = m^2 F_{AD}^2 D_{ivo}^2 \sigma_v. \tag{4.10}$$

The digital representation of the third controller input, the output current i_o, is scaled first by D_{cur} from the current sense circuit, and then by F_{AD} from A/D sampling.

$$i_{od} = F_{AD} D_{cur} i_o. \tag{4.11}$$

Since the reference current and the current loop PI accumulator are of the same units ($Amps$) as the output current, then their digital representations are scaled identically.

$$I_{od} = F_{AD} D_{cur} I_o. \tag{4.12}$$

$$\sigma_{id} = F_{AD} D_{cur} \sigma_i. \tag{4.13}$$

Controller Gains and Parameter Limits

The following manipulations result in digital representations of the discrete time PI equations. Rewriting the PI equation (4.2) for the voltage loop

$$k[n] = \frac{C}{T_L V^2}(h_1(V_o^2 - v_o^2) + h_2\sigma_v) + \frac{2}{V^2}v_o i_o, \tag{4.14}$$

substituting for v_o^2, V_o^2, V^2, and σ_v in terms of their digital equivalents, and adding a scaling factor G_{dig} yields

$$k[n] = \frac{C}{T_L}\frac{D_{ivi}^2 F_{AD}^2 G_{dig}}{V_{id}^2 G_{dig}}(h_1\frac{(V_{od}^2 - v_{od}^2)}{m^2 D_{ivo}^2 F_{AD}^2} + h_2\frac{\sigma_{vd}}{m^2 D_{ivo}^2 F_{AD}^2}) + \frac{2}{V^2}v_o i_o. \tag{4.15}$$

The next equation results from substituting digital versions of v_o and i_o into (4.15). Additionally, the command k is scaled by $k_d = \frac{4095}{k_{max}}$, so that the result of the PI equation is the digital command k_d.

$$\begin{aligned} k_d[n] = {} & \frac{4095}{k_{max}}\frac{C}{T_L}\frac{D_{ivi}^2 G_{dig} F_{AD}^2}{m^2 D_{ivo}^2 F_{AD}^2}\frac{1}{V_d^2 G_{dig}}(h_1(V_{od}^2 - v_{od}^2) + h_2\sigma_{vd}) + \\ & \frac{4095}{k_{max}}\frac{2D_{ivi}^2 F_{AD}^2}{mD_{ivo}D_{cur}F_{AD}^2}\frac{v_{od}i_{od}}{V_d^2} \end{aligned} \tag{4.16}$$

In order for the digital voltage loop PI equation to be

$$k_d[n] = \frac{1}{V_d^2 G_{dig}}(h_{1d}(V_{od}^2 - v_{od}^2) + h_{2d}\sigma_{vd}) + h_{pd}\frac{v_{od}i_{od}}{V_d^2}, \tag{4.17}$$

then the digital gains for the voltage loop control, from (4.16), must be

$$h_{1d,2d} = \frac{4095}{k_{max}}\frac{C}{T_L}\frac{D_{ivi}^2 G_{dig}}{m^2 D_{ivo}^2}h_{1,2} \tag{4.18}$$

and

$$h_{pd} = \frac{4095}{k_{max}}\frac{2D_{ivi}^2}{mD_{ivo}D_{cur}} \tag{4.19}$$

Rewriting the current loop PI equation (4.3)

$$V_o[N] = h_3(I_o - i_o) + h_4\sigma_i \tag{4.20}$$

and substituting for V_o, I_o, i_o, and σ_i in terms of their digital equivalents yields

$$V_{od}[N] = \frac{F_{AD}mD_{ivo}}{F_{AD}D_{cur}}(h_3(I_{od} - i_{od}) + h_4\sigma_{id}). \tag{4.21}$$

Parameter	Value	Units
C	470	μF
T_L	8.333	ms
V	168	V
V_{id}	261	V
h_1	.1	-
h_2	.0025	-
h_3	1958	Ω
h_4	2150	Ω
h_{1d}	152	-
h_{2d}	973/256	-
h_{3d}	4	Ω
h_{4d}	4497/1024	Ω
h_{pd}	220/1	-
G_{dig}	.5	-
σ_{vdmax}	36762754	V
σ_{idmax}	761	V
k_{max}	.00263	$1/\Omega$
k_{maxd}	4095	$1/\Omega$
D_{ivo}	$9.75 * 10^{-3}$	V/V
D_{ivi}	$7.6 * 10^{-3}$	V/V
D_{cur}	20	A/A
F_{AD}	204.6	V/V

Table 4.1: Software parameter values.

In order for the digital current loop PI equation to be

$$V_{od}[N] = h_{3d}(I_{od} - i_{od}) + h_{4d}\sigma_{id}, \tag{4.22}$$

then the digital gains for the current loop, from (4.21), must be

$$h_{3d,4d} = \frac{mD_{ivo}}{D_{cur}} h_{3,4}. \tag{4.23}$$

Example Parameters

Table 4.1 contains parameter values in the implemented prototype. Appendix A explains how to calculate the maximum values of the digital accumulators.

4.2.3 Other Controller Features

This section describes the software implementation of the additional control features discussed in Section 2.2.4. The software performs accumulator anti-windup, soft start, and steady state band command averaging for both the voltage and the current loops.

Anti-windup

To eliminate the possibility of accumulator windup, the voltage loop digital accumulator σ_{vd} ceases accumulating squared voltage errors when the command from the PI calculation exceeds 4095. Since a too high current loop voltage reference command leads to a too high current loop boost converter input current command, then it is sufficient to cease accumulation only in the voltage loop control.

Soft Start

Soft start within the software occurs in two stages. First, when the boost converter output voltage lies below the resolution enhancement mapped region, the digital controller increases the input current command monotonically in an open loop setting (PI disabled). When the output voltage reaches the mapped region, then the current reference is increased until a target is reached. This second part of soft start occurs in a closed loop setting (PI enabled). The amount by which the reference is periodically augmented was experimentally chosen so that the linear increase in output current does not abruptly change during the transition from open to closed loop.

Steady State Bands

The software for both the current and the voltage loops implement steady state band averaging, and operates as shown in Figure 4-11. A counter for both loops is incremented when the respective output lies within its steady state band. When the counter reaches the predetermined minimum number N, then the average of the last N commands is used for the new command. The counter is reset and the process begins anew when the output falls outside the band. There is a small cushion region between being "in" and being "out" of the steady state band to effect a hysteresis in case of small, noisy output deviations.

To choose steady state bands, first an allowable error region is assumed. Setting this error region to be $\pm.5\%$ and the current reference I_{od} to be 387, for instance, yields the

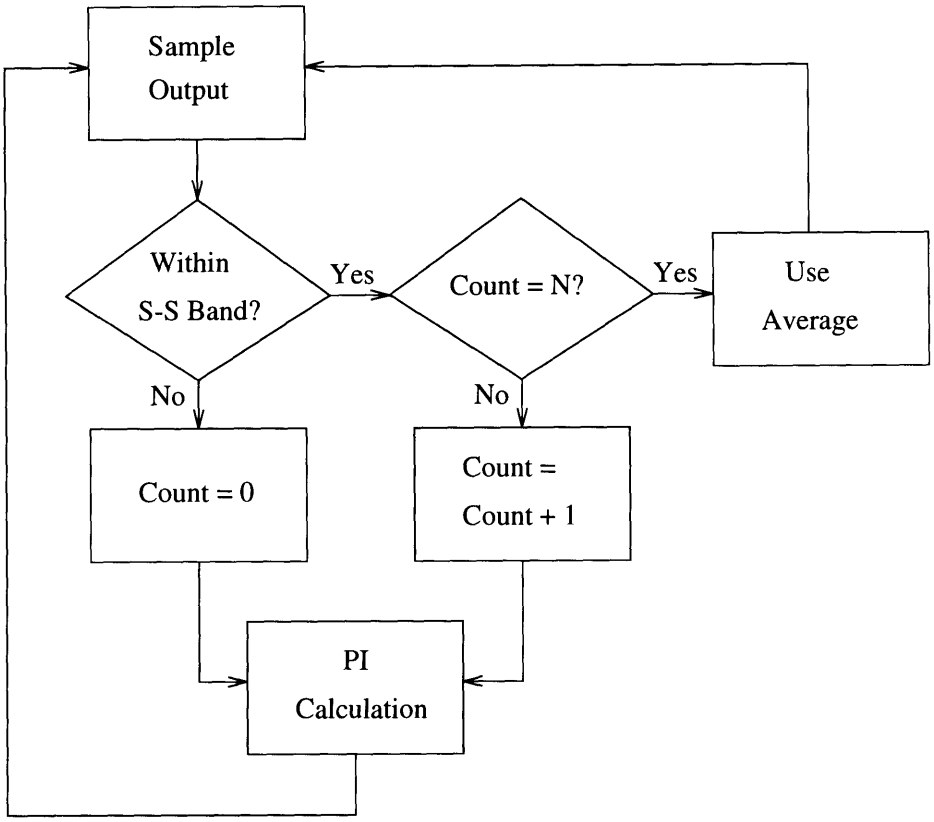

Figure 4-11: Implementation of steady state band averaging.

steady state band values 385 and 389. When the current loop provides a voltage reference to the voltage loop, it must also provide voltage steady state band values. Allowing the error region in the voltage loop to be ±.78%, then the steady state bands are taken to be $V_{od} \pm (V_{od}/128)$. Dividing by 128 is an easy shift to the right by 7 with integer arithmetic.

Integer Arithmetic

Since divides are time intensive, and since only integers are represented inside the microcontroller, it is pragmatic to fragment calculations. For instance, if the current loop accumulator digital gain h_{4d} is 4.392, then an efficient calculation is

$$A = 4.392 * \sigma_{id} = \frac{4497}{2^{10}} * \sigma_{id}. \tag{4.24}$$

Dividing by powers of 2 is accomplished by shifts to the right rather than divide instructions. Care is taken to ensure that arithmetic does not exceed $2^{31} - 1$ or -2^{32}, the limits for 32 bit signed integers; at the same time, gains are tailored so that calculations use as much bit resolution as possible for maximum accuracy.

4.3 Transformer Subsystem Hardware

This section explains the hardware features of the inductive coupling subsystem of Figure 1-1. The major components, referring to Figure 3-7, are the high frequency DC/AC half-bridge inverter, gapped core transformer, and rectifier. First, the digital circuit that provides a switching pattern for the MOSFETs is discussed. Second, the MOSFET gate drive circuits are presented. Finally, specifications on the transformer in the prototype are given.

4.3.1 Digital Switching Pattern Generator

The circuit depicted in Figure 4-12 was taken from [27], and is capable of producing 7 periodic signals. Counters increment through the low eight bits of the EPROM's 15 bit address space. The top seven bits are tied to ground. While the EPROM features 8 outputs, one output resets the counting process by providing a CLR strobe to the counters. In the half bridge case, only two transistors require switching. Therefore, two output bits are toggled as shown

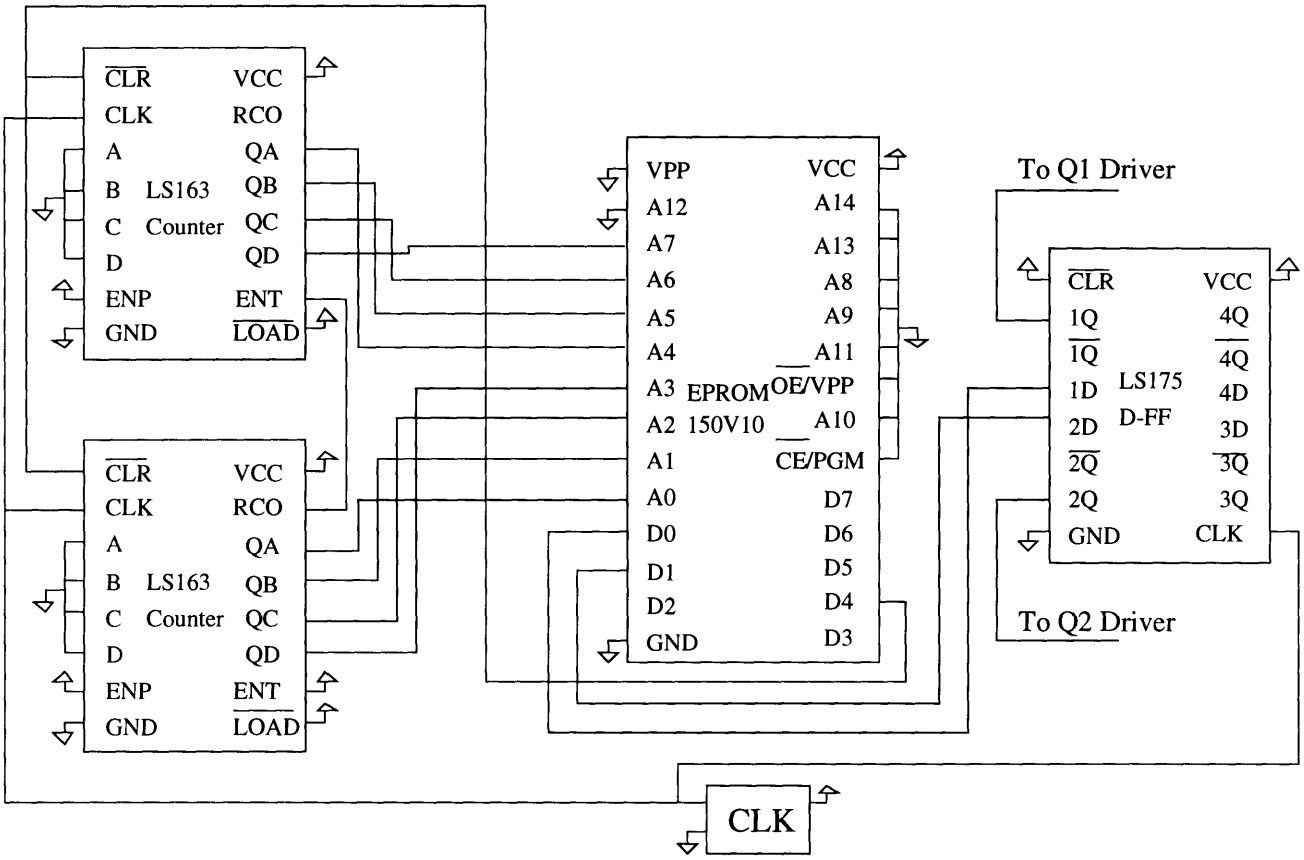

Figure 4-12: Digital circuit to provide FET switching signals.

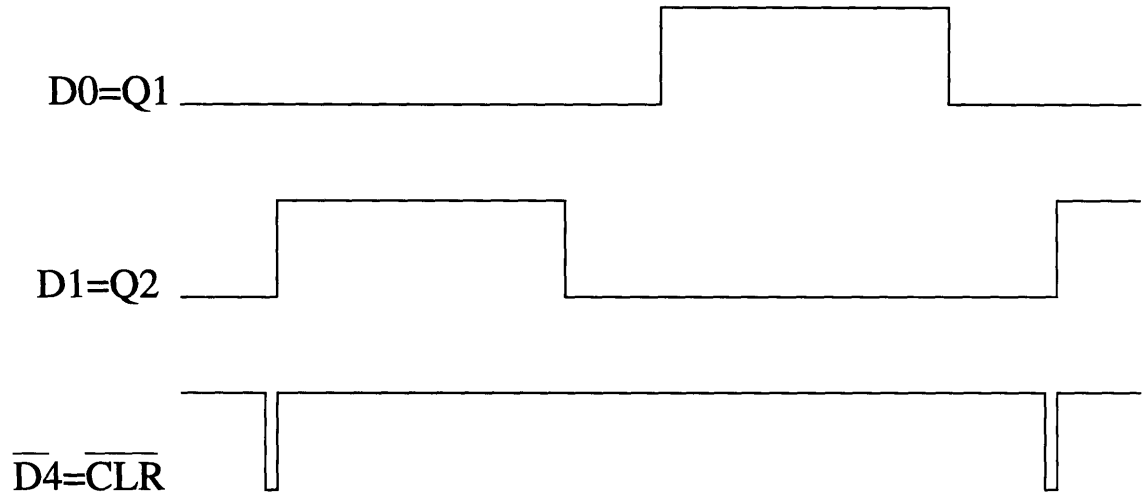

Figure 4-13: Switching signals from EPROM output bits, plus counter active low CLR signal.

in Figure 4-13. The two chief conditions that must prevail are

1) both outputs must never both represent their respective transistor being on to avoid shoot through, and

2) there must be enough time when both output bits are "off" to enable the energy recovery of the lossless switching scheme to occur.

In the prototype, a crystal oscillates at 10MHz with 200 addresses accessed before the counters are reset. The frequency is thus $\frac{10*10^6}{200} = 50kHz$. Using only 200 samples in one period results in less time-domain granularity in the output bits than more samples would support. However, since the off-time of both transistors was as high as 15 crystal cycles in experiments, and since eight address counter output bits provide only 256 accessible addresses, and since the EPROM is not faster than 10MHz, the crystal frequency and number of samples in a period are suitable for the prototype.

The immediately obvious advantage of this implementation is the flexibility it affords. A completely new switching scheme can be tested simply by reprogramming an EPROM. The frequency of the DC / AC inversion can be changed by replacing the crystal oscillator, or by including a different number of samples in the cyclical EPROM byte sequence.

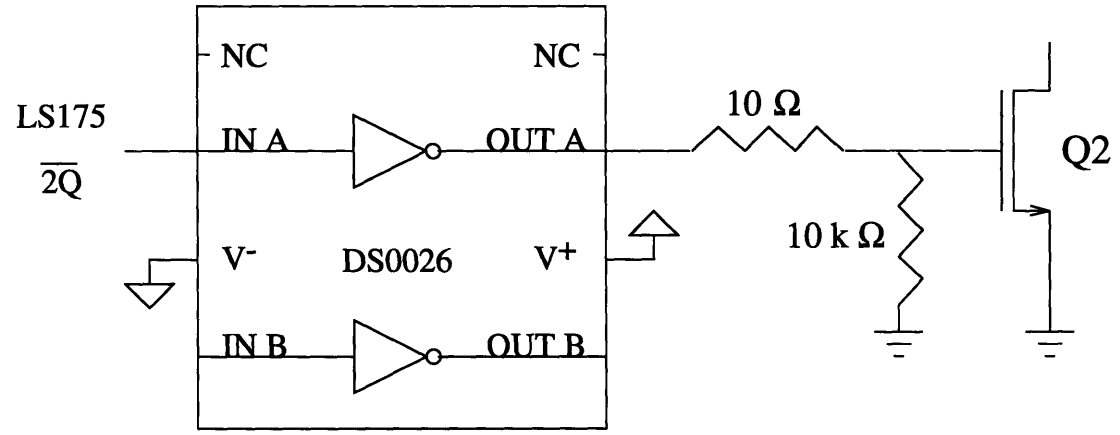

Figure 4-14: Low side transistor gate driving circuit.

4.3.2 FET Gate Drivers

A MOSFET will turn on when the gate-to-source voltage is greater than a threshold voltage ($V_{GS} > V_T$). Special purpose devices are used as gate drivers to ensure fast charging and discharging of the gate space charge layer. Applying a high enough voltage to the gate of the low side transistor of Figure 3-1 is easy, since the source is simply ground. Driving up the high side transistor gate voltage is more difficult because the source voltage of the transistor is variable. The driver circuits are discussed in this section.

Low Side Driver

The National Semiconductor part DS0026 serves as the half-bridge low side MOSFET gate driver [39]. The output gate drive is inversely mapped from the TTL input 0 or 5 Volts to an output 14 or 0 Volts. The output is applied through a resistor divider to the base of $Q2$, as shown in Figure 4-14.

High Side Driver

Unitrode provides a circuit with a floating ground that functions as a high-side driver [40]. The UC3724 emits a modulated signal based on a digital input referenced to ground. This signal is impressed on the primary of a small transformer whose secondary is referenced to the source of the inverter high-side FET. The gate-to-source voltage V_{GS} is forced low or high by the UC3725 based on the modulated signal on the transformer secondary. Power must be supplied to the UC3724, but power is delivered to the UC3725 through the transformer from the UC3724. The circuit can be found in Appendix D.

4.3.3 Transformer Specification

The battery load in the secondary rectifier may typically require 14 Volts DC. For the half bridge configuration of Figure 3-1, the primary voltage swings between plus and minus 150 Volts for a $300V$ DC input. Thus the turns ratio $N_p : N_s$ should be 10.7 for the voltage on the ~~primary~~ to swing between plus and minus $14V$. The values $N_p = 250$ and $N_s = 23$ are picked with ensuring the core does not saturate in mind. Measuring the inductance of the primary while shorting the secondary indicated a leakage inductance of $L_{lk} = 5.26mH$ at $50kHz$. The measured magnetizing inductance at $50kHz$ was $L_\mu = 33.55mH$.

Letting the primary voltage be 150V for half a cycle at $50kHz$, the flux linkage reaches a maximum $V_m \frac{1}{2} \frac{1}{50000}$, where V_m is the voltage across the magnetizing current from the impedance division of the primary voltage. Then $V_m = 150 \frac{L_\mu}{L_\mu + L_{lk}} = 129.7V$, and $\lambda_{max} = 1.30mWb$. Using $\lambda = Li$, the maximum value of magnetizing current is therefore $I_{Pkmag} = .0387Amps$, which is a safe value that leads to thin copper windings.

Chapter 5

Experimental Results

Prototypes for the charging current controller and the inductively coupled interface were built and tested. The comparison of experimental results with expected results based on theory and simulation demonstrates the validity and usefulness of the ideas presented in the previous discussion.

5.1 Unity Power Factor

While the match between simulation and experiment indicates a high probability that the models represent well the system, it is vital that the original assumption that the boost converter input current is kv_{in} be verified. The typical uncorrected input current to a full bridge rectifier such as the one in the boost converter from Figure 1-2 is shown again for convenience in Fig. 5-1. Figure 5-2 shows the power factor corrected input current while running only the voltage loop in steady state at about 350V output. It is basically sinusoidal, which corroborates the assumption that $i_{in} = kv_{in}$ that was necessary for the theoretical discussion in Chapter 2. Additionally, it can be observed that the second negative slope zero crossing of the input current in Fig. 5-2 occurs around 16 or 17 ms after the first, which agrees with one cycle being $\frac{1}{60Hz} = 16.7ms$.

Boost Converter Efficiency

Comparing the boost converter input power to output power reveals that most of the power drawn from the utility is dissipated across the load. Input power is

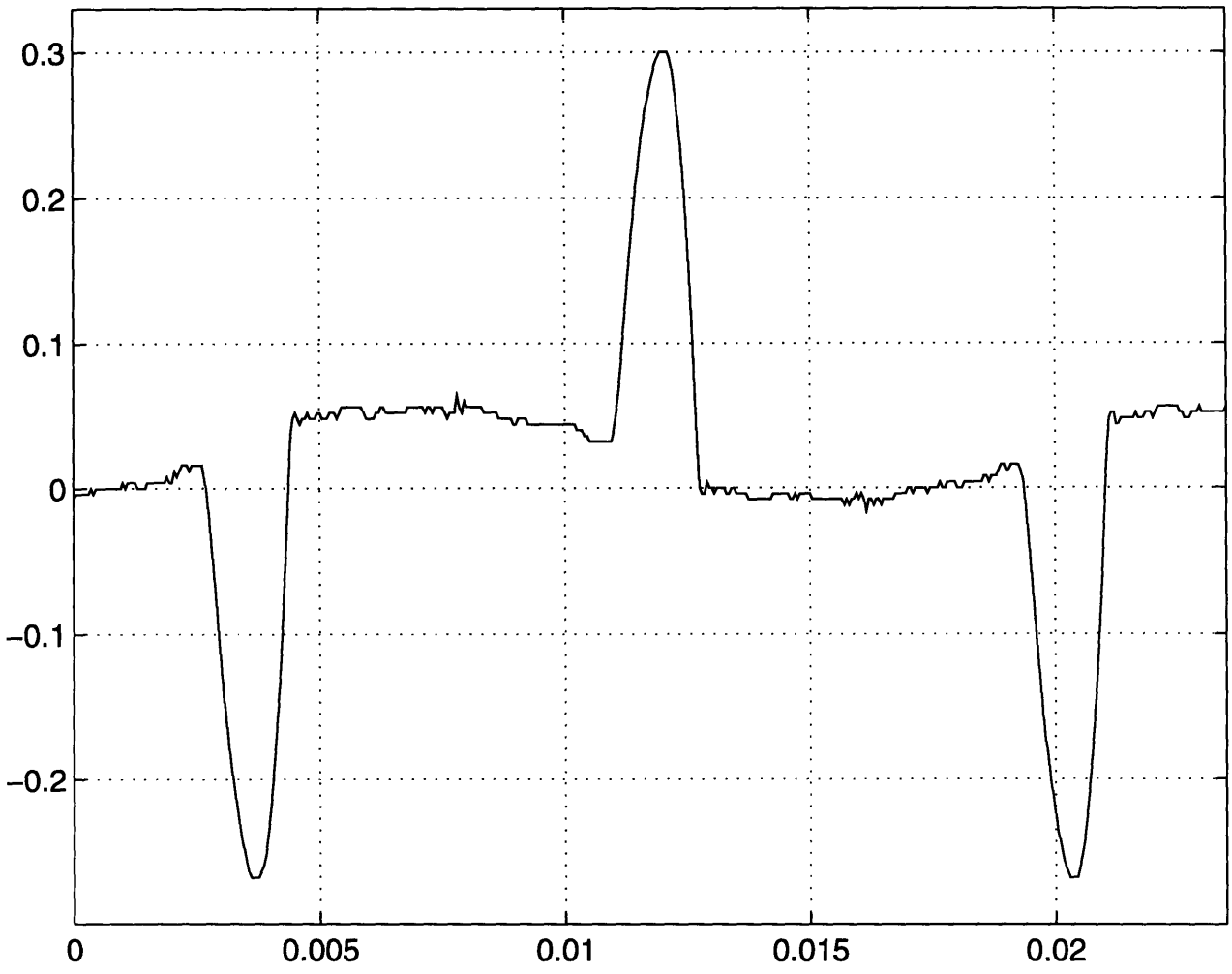

Figure 5-1: Uncorrected input current - one period.

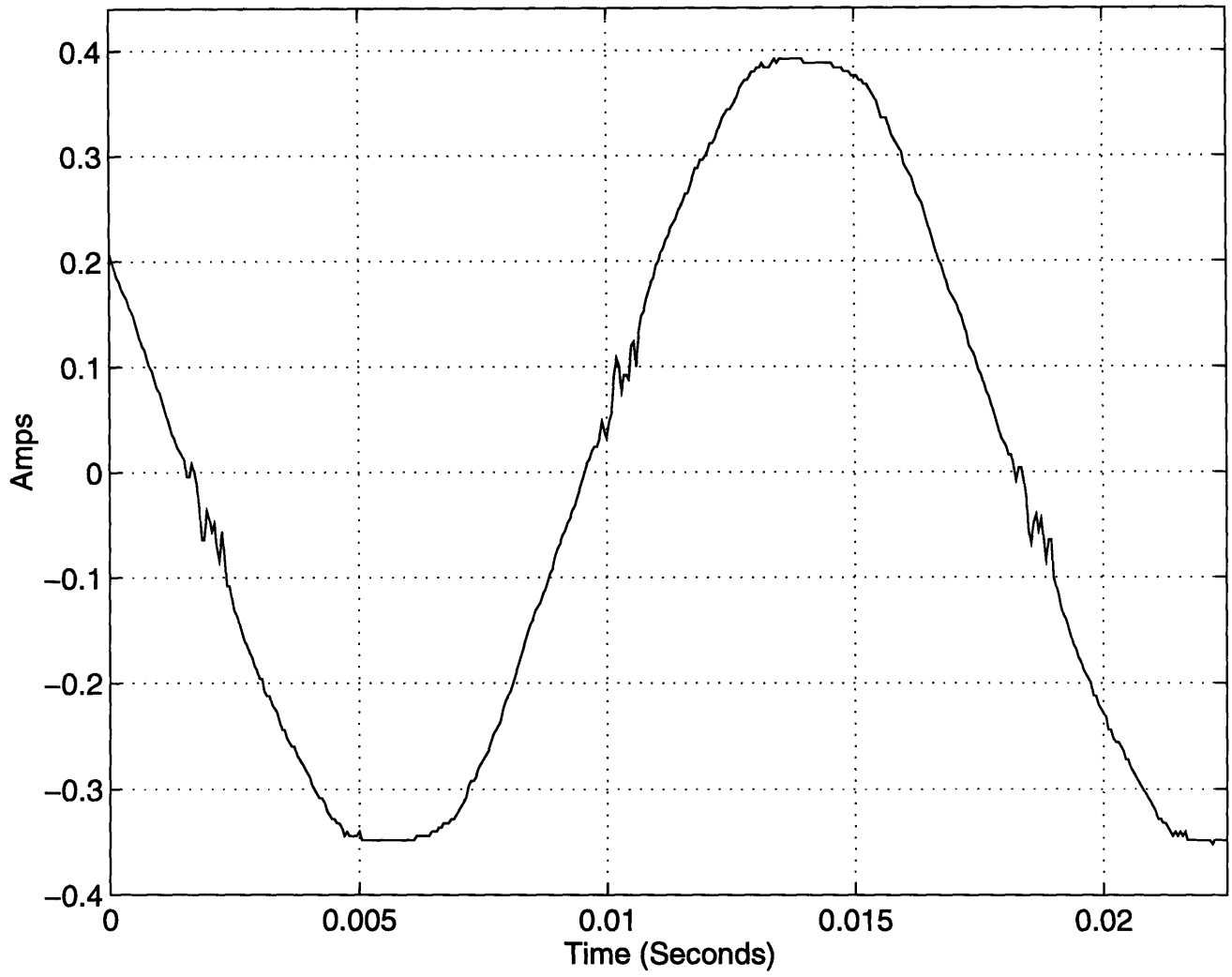

Figure 5-2: Sinusoidal input current - one period.

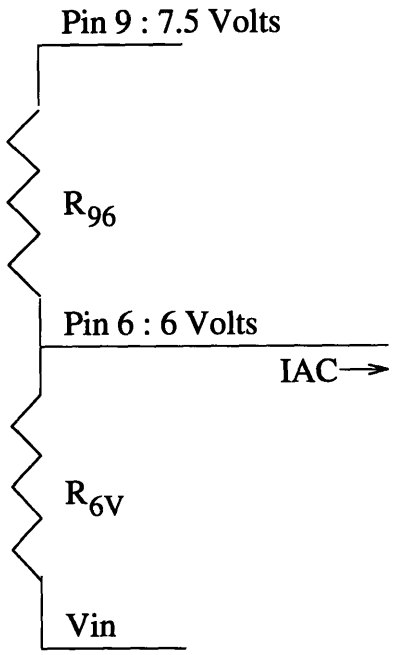

Figure 5-3: Resistors for adjusting current input to UC3854 pin 6.

$$I_{rms} * V_{rms} = \frac{(.442)(168)}{2} = 37.1Watts. \tag{5.1}$$

For a load resistor $3.91k\Omega$ and a boost converter output voltage $380V$, load power is

$$\frac{V^2}{R} = \frac{380^2}{3910} = 36.9Watts. \tag{5.2}$$

The ratio of input power to output power is .995. However, the system is not 99.5% efficient because the microprocessor, the analog hardware, and the UC3854 gate MOSFET gate drive all require power from the three DC power supplies.

Crossover Distortion

The crossover "distortion" in Figure 5-2 is the result of a slight resistor mismatch in the inner loop analog current controller. Figure 5-3 shows the circuitry surrounding the UC3854, pins 6 and 9. The signal V_{in} is the scaled rectified input voltage. According to the Unitrode specification, when V_{in} is zero, the UC3854 pin 6 input current I_{AC} should be zero. When V_{in} is its max value, in this case $5V$, I_{AC} should be no greater than $400\mu A$ [11]. For I_{AC} to be zero for $V_{in} = 0$, the resistor ratio should, from nodal analysis, be $R_{96} : R_{6V} = 1 : 4$.

In the prototype, $R_{96} = 4.41k\Omega$ and $R_{6V} = 14.34k\Omega$. Then when $V_{in} = 5V$, $I_{AC} = 270\mu A$, meeting the specification. When $V_{in} = 0V$, $I_{AC} = -78.3\mu A$. The resistor $R_{96} : R_{6V}$ in the prototype is not 1 : 4, but experiments showed that without the use of a potentiometer, the chosen values minimized crossover distortion.

5.2 Charging Current Control System

The complete closed loop current control with feed forward design was simulated and tested. Data were taken for both the voltage loop only and for the entire current loop system.

5.2.1 Experimental Setup

The AC utility input was fed through a VARIAC (variable AC transformer) to allow for adjustment of the input voltage [34]. A load resistor of $3.91k\Omega$ was used on the boost converter output. The construction of the charger utilized three boards. The boost converter was soldered on a PC board designed by Steven Shaw. Several voltages to be monitored were taken with twisted pair connections to another board containing the Unitrode current loop controller, several isolation amplifiers, multiplying DAC, resolution mapping circuit, current sensing circuit, and filter components. Connected by standoffs, the circuit boards were placed within a grounded testing station. Twisted pair wire wrapping connected the microprocessor A/D ports and HSO lines to the analog boards. The three DC power supplies were available at the testing station.

Three main goals pervaded four experiments. The first objective was to compare the output current and voltage reference step responses to simulations, which are based on models. The second objective was to observe that the system can track a current profile. The third objective was to validate the execution of soft start. The first experiment tested the voltage loop against simulation for a single step response upwards and downwards. A single current reference step change upwards and downwards comprise the next experiment. Another features an oscillating current reference. The fourth monitors the response to a slowly decreasing current reference.

5.2.2 Voltage and Current Loop Simulation

All four experiments were simulated using MATLAB. The simulations did not model the system as a state space equation, assume an initial condition, and carry out subsequent state transitions. Rather, the MATLAB programs attempted to predict exactly the real responses by modeling the physical equations that lead to theoretical state space descriptions, and then duplicating the software. The three current control MATLAB programs are identical, with the exception of current profile entry. Since the real system operates with a timer interrupt, the simulations must provide time indexes for the iterations for comparison.

The simulations incorporate not only the mechanics of the physical system, PI control, and the changing reference, but they also predict soft start response. The command *floor* is used to model integer arithmetic. Figure 5-4 shows the voltage loop stepping reference versus simulation. Figures 5-5 and 5-6 show the expected step, oscillatory, and ramping responses along with the references. The dotted lines are the references, while the solid lines are the predicted responses. The reference during the half of startup that is open loop (while the output voltage is less than the mapping voltage) contains no information.

5.2.3 Voltage and Current Loop Experimental Results

The prototype performs steady state band control admirably; but since the simulation did not incorporate that functionality, this comparison uses data generated without steady state band control in the prototype. Figure 5-7 presents the voltage loop experiment and simulation. The output voltage loop curve matches the simulation almost perfectly. The delay model for the voltage loop is valid, and the current loop is expected to operate as simulated. Figures 5-8, 5-9, and 5-10 demonstrate the incredible accordance with prediction in which the prototype behaves. Dotted lines are simulation curves. Figures 5-11, 5-12, 5-13, and 5-14 show the reference (dotted / dashed line), the simulation (dashed line), and the experimental curve (solid line) for all four experiments.

The MATLAB and C code can be found in Appendix E.

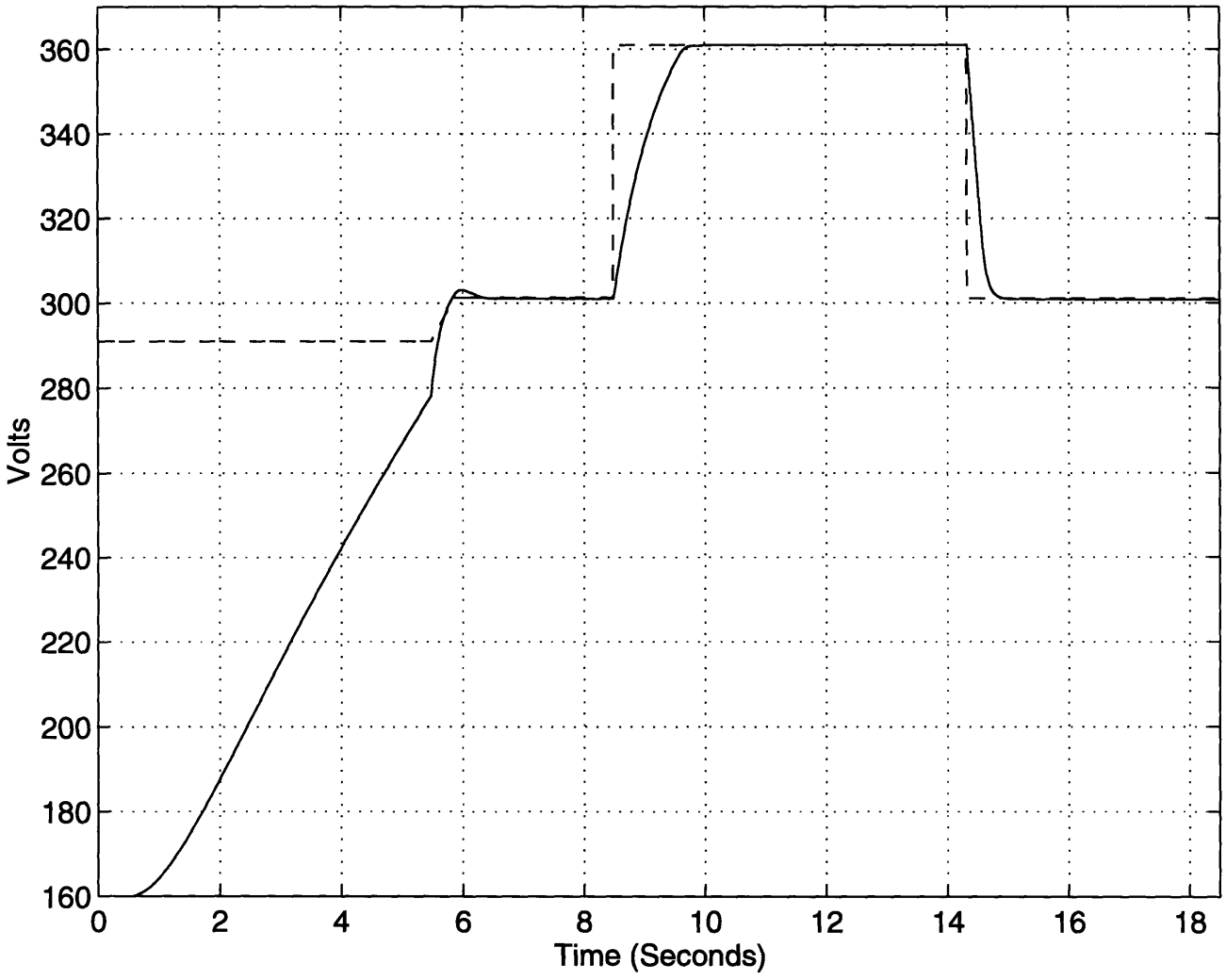

Figure 5-4: Simulated voltage single step response plus reference.

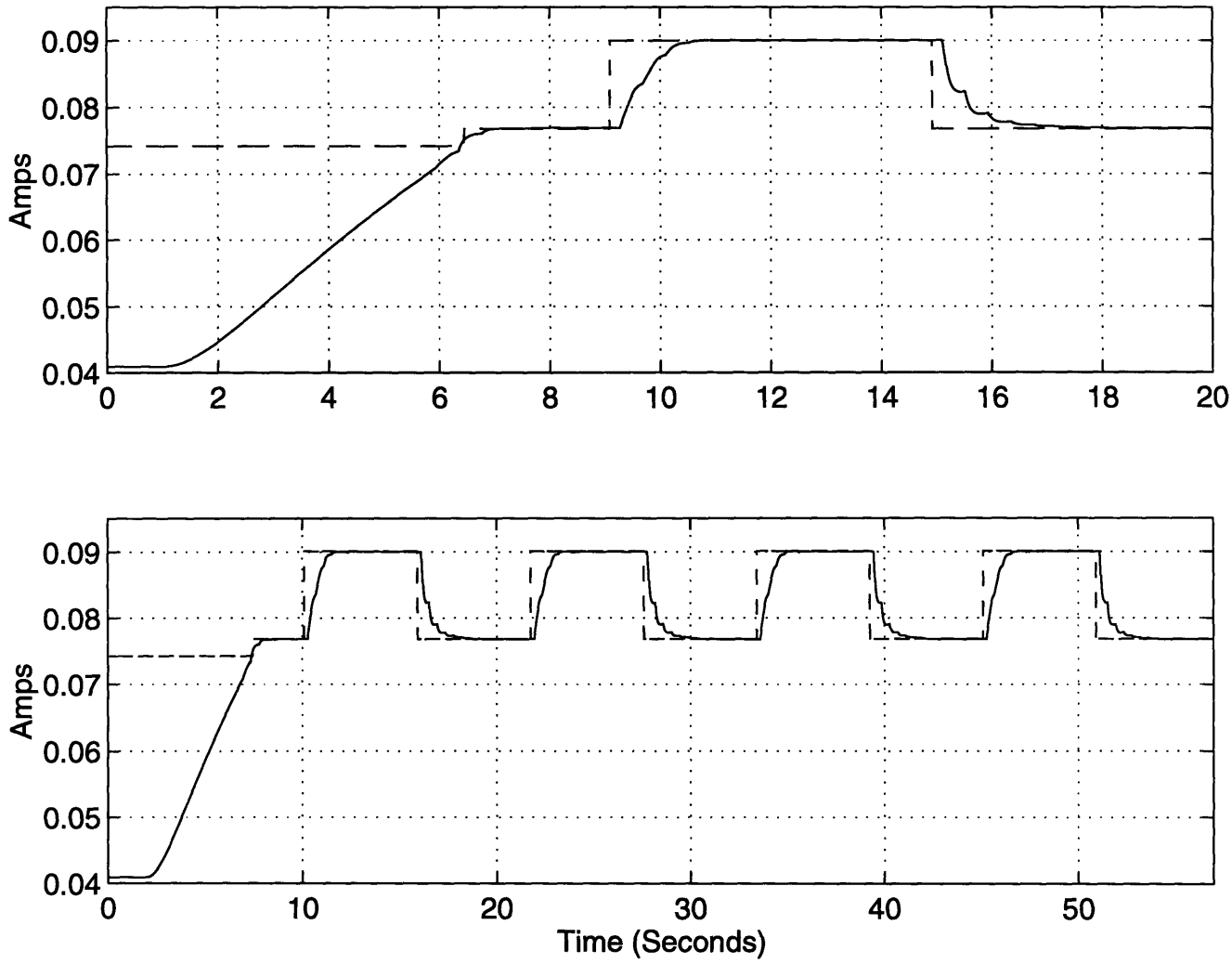

Figure 5-5: Simulated charging current step responses plus references.

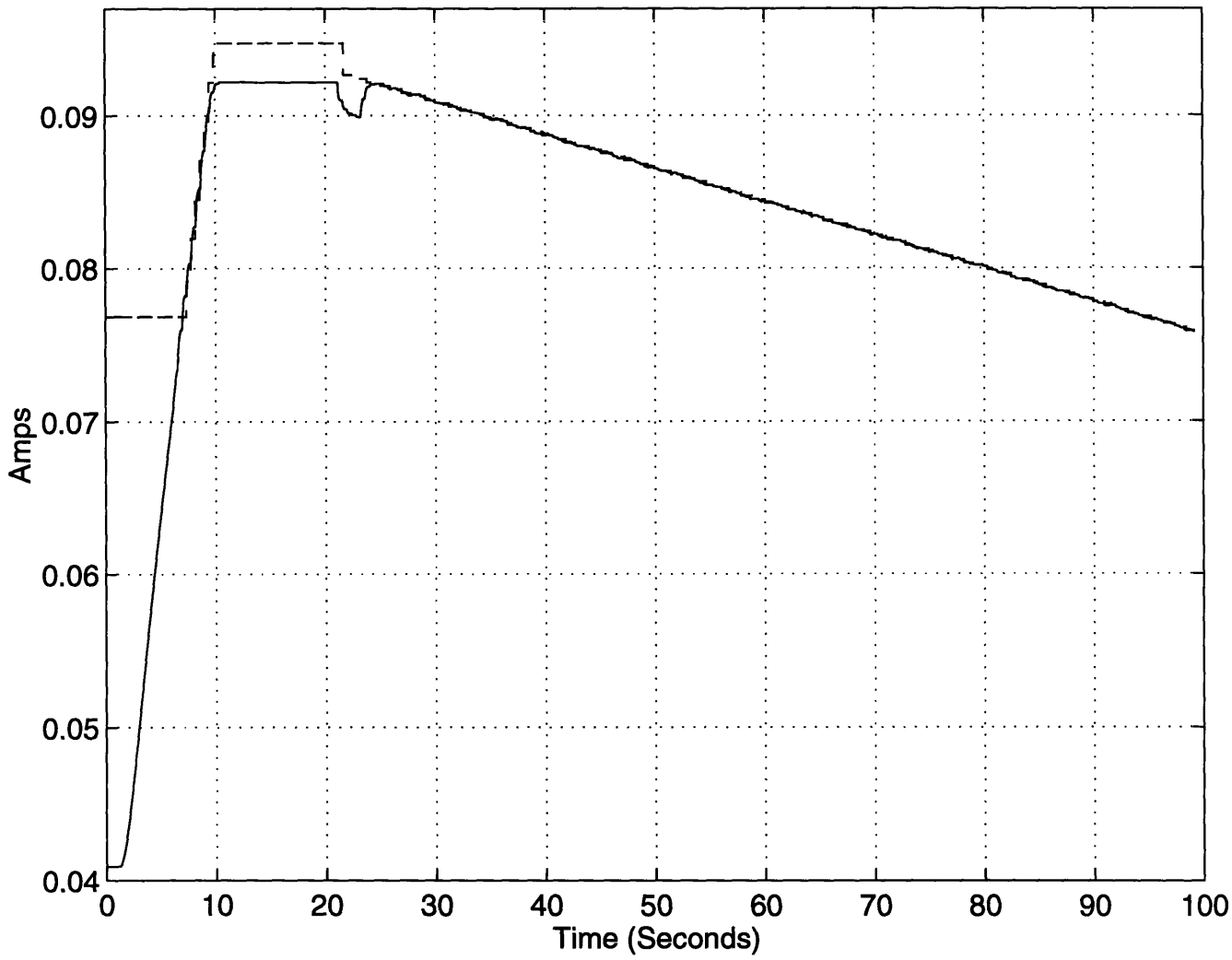

Figure 5-6: Simulated charging current ramp response plus reference.

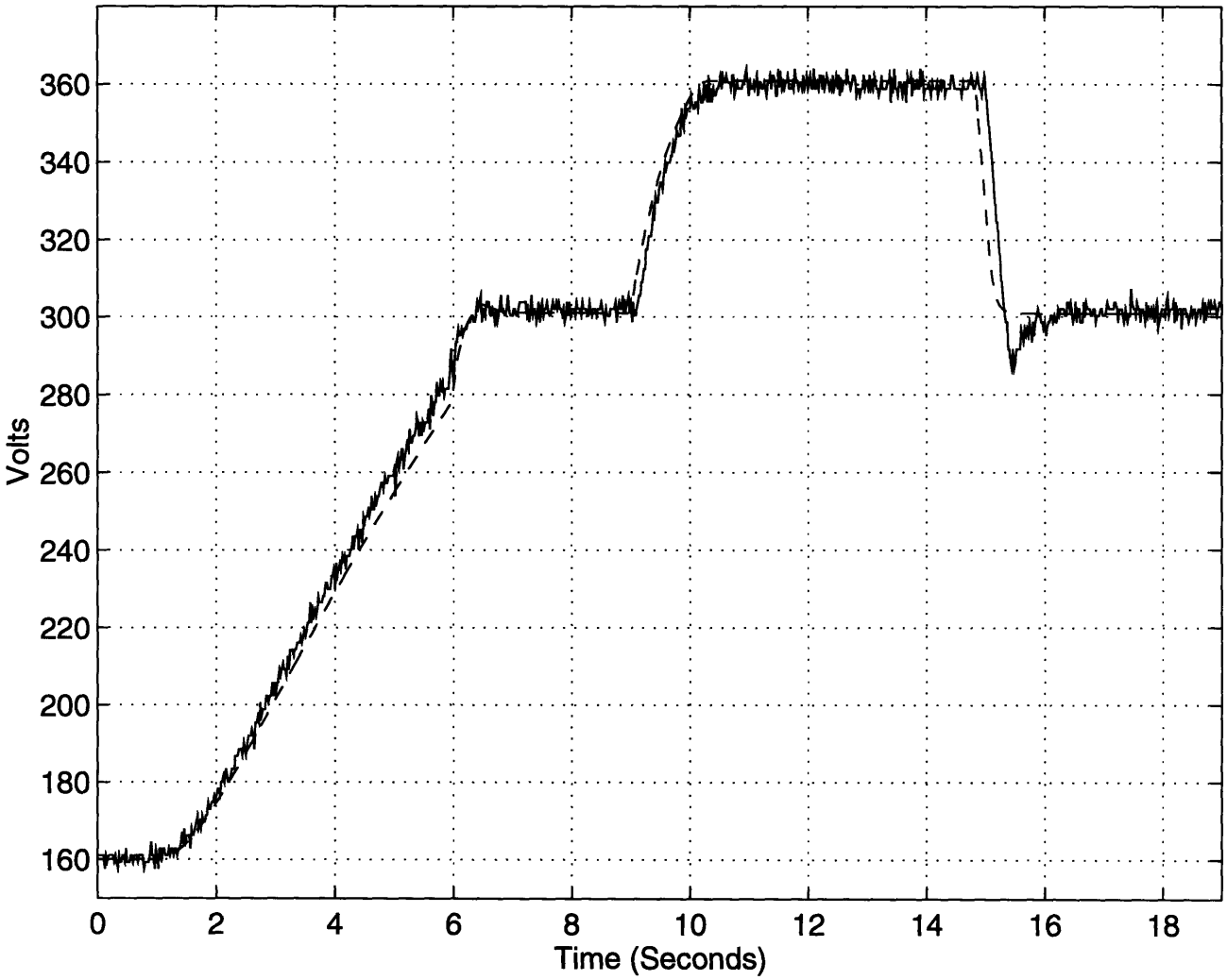

Figure 5-7: Simulated and experimental voltage single step responses.

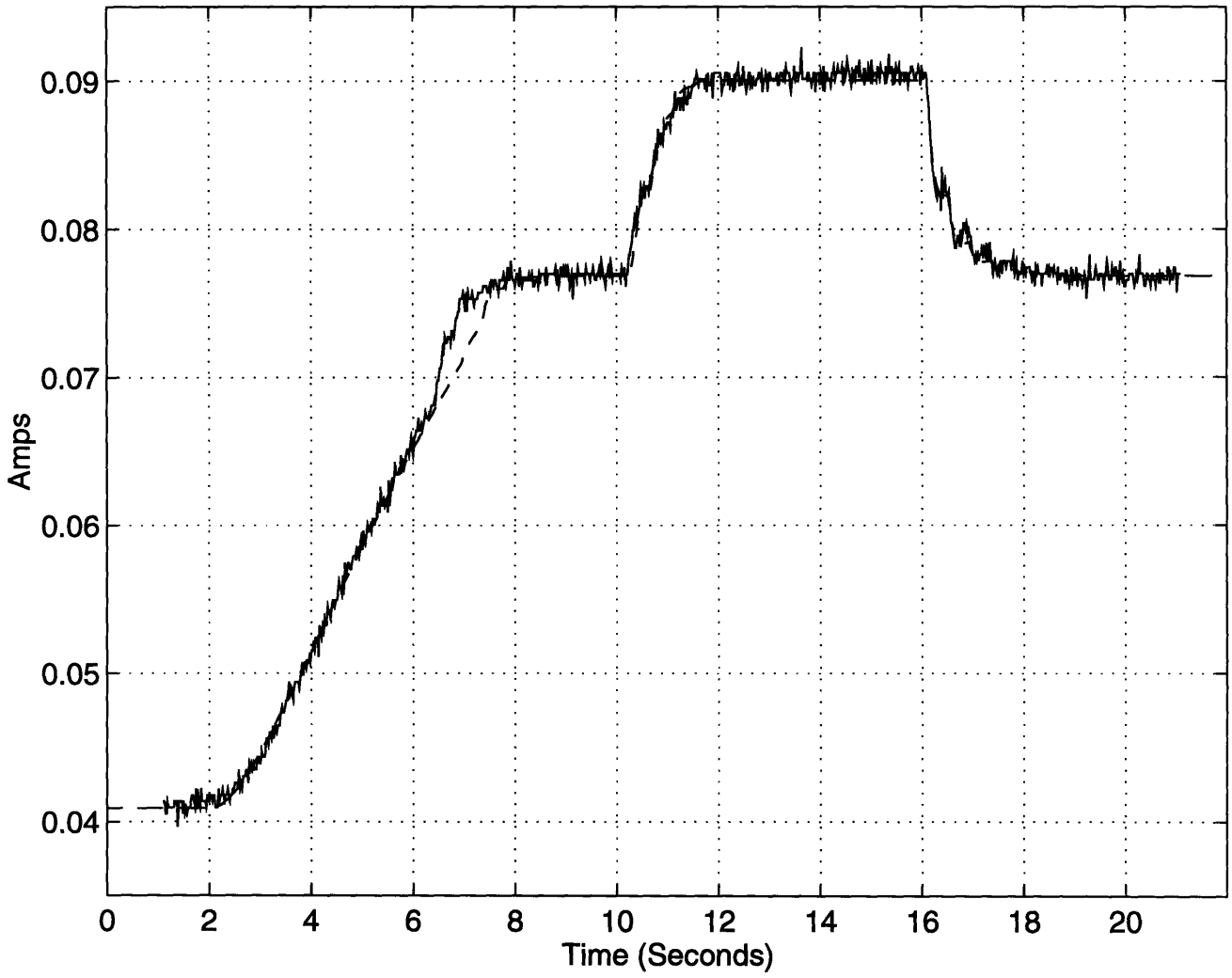

Figure 5-8: Simulated and experimental current single step responses.

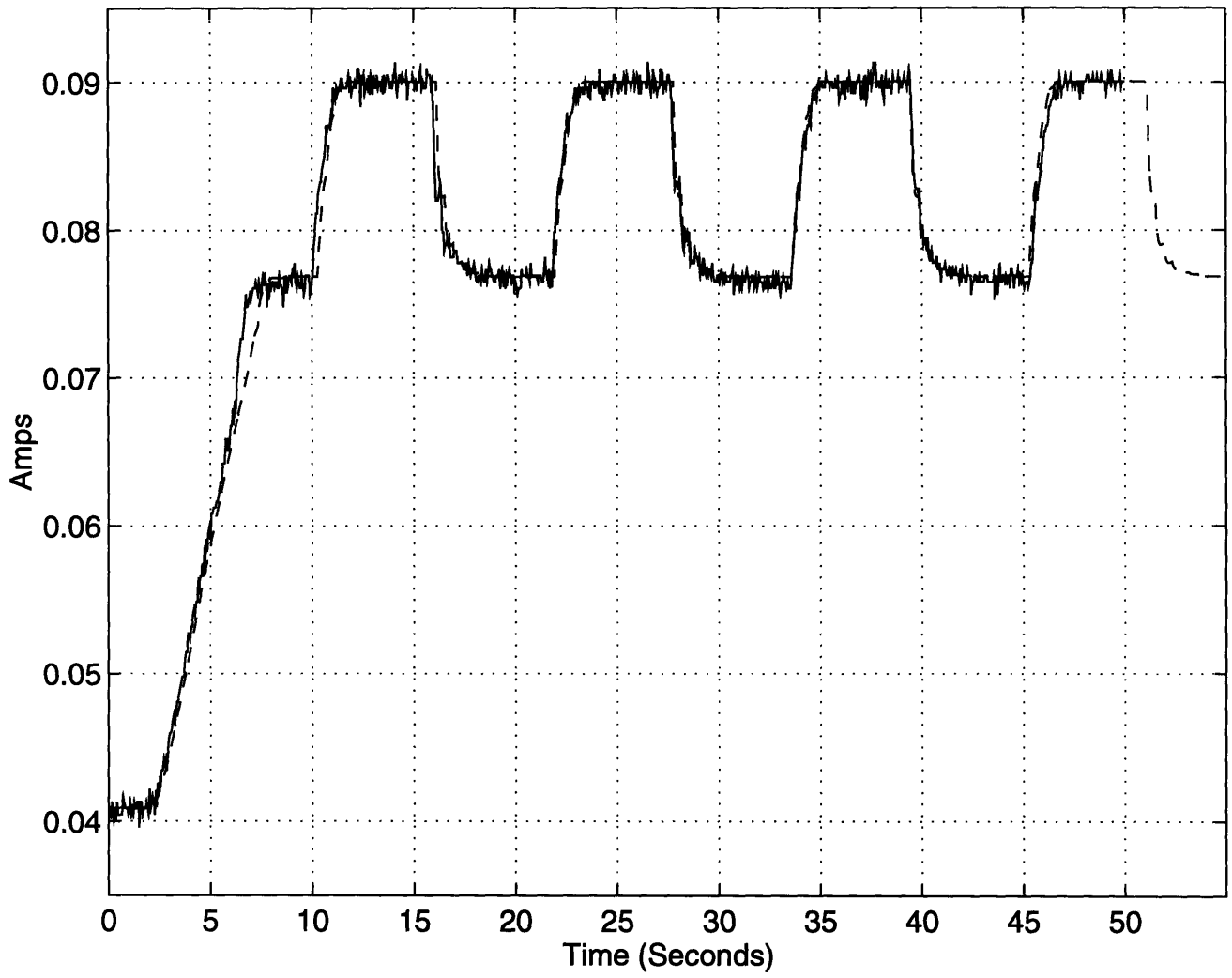

Figure 5-9: Simulated and experimental current oscillatory step responses.

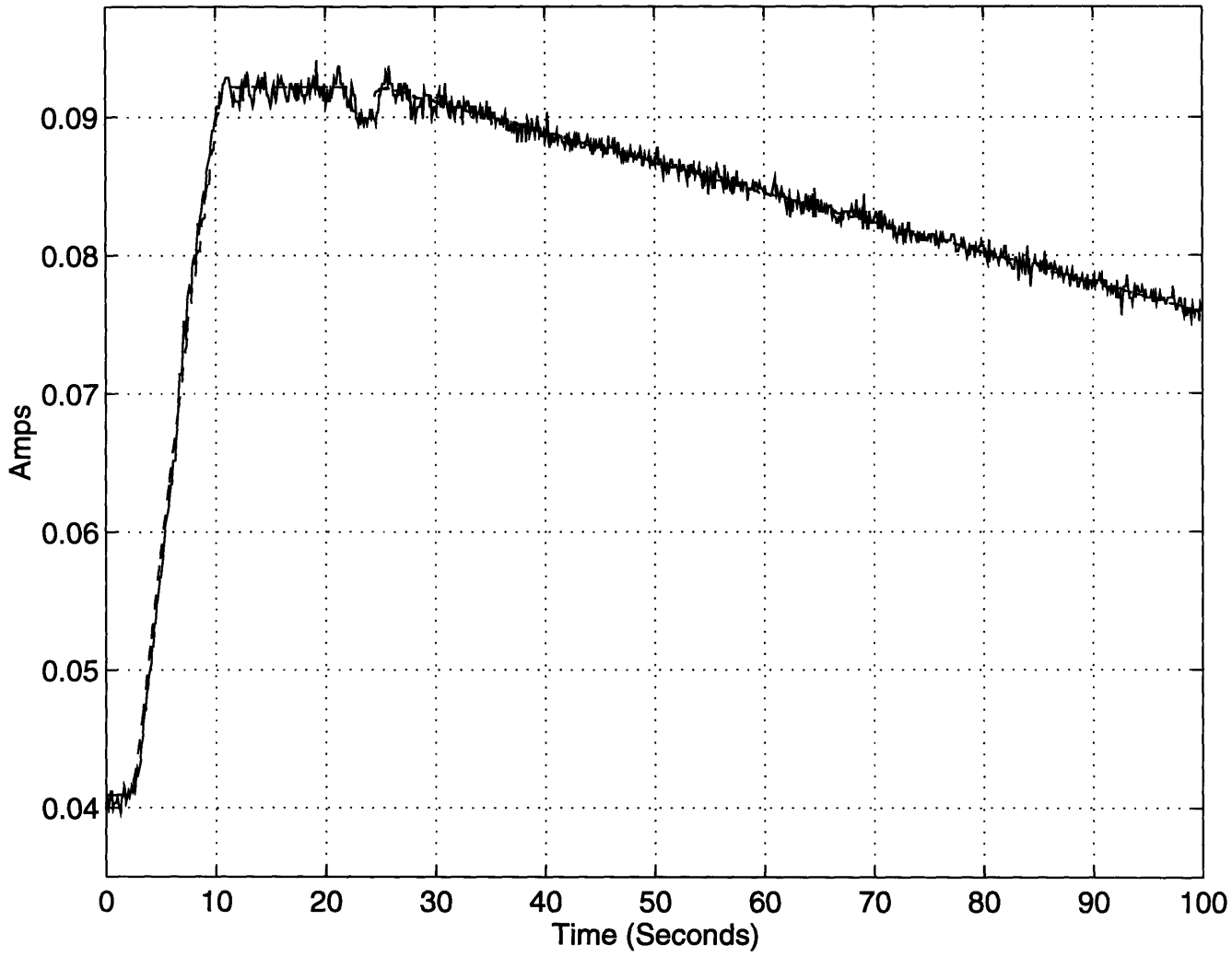

Figure 5-10: Simulated and experimental current ramp responses.

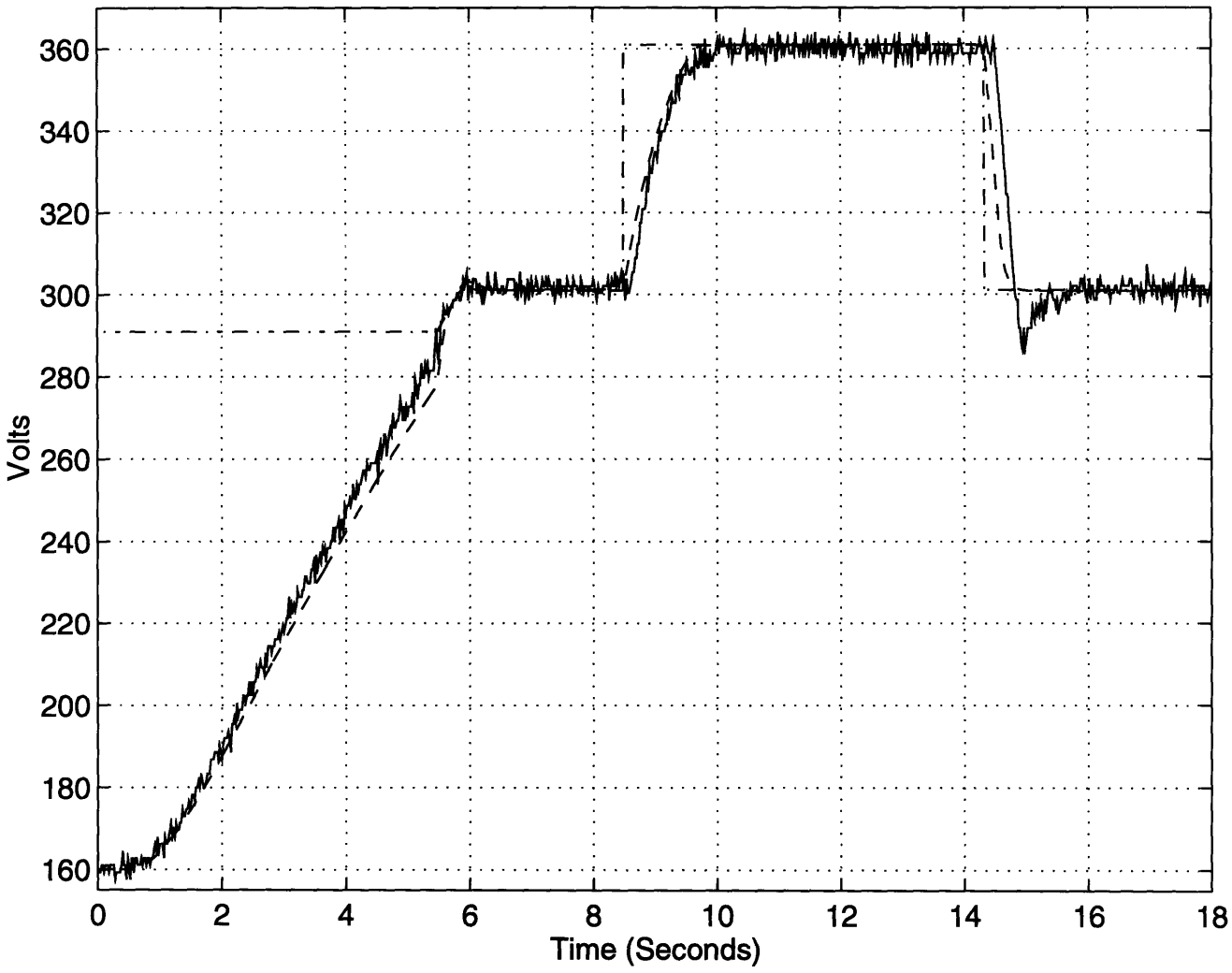

Figure 5-11: Simulated and experimental voltage single step responses plus reference.

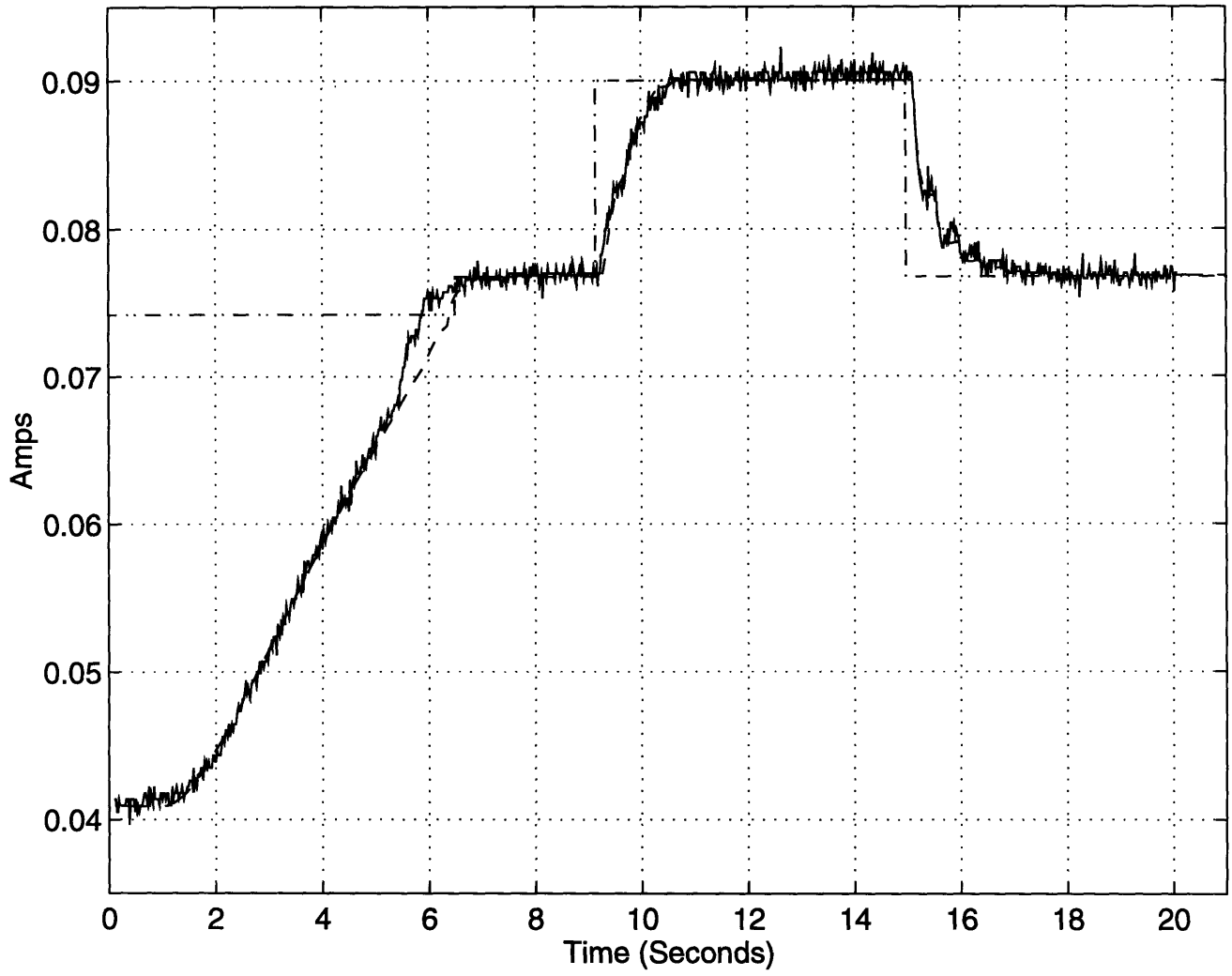

Figure 5-12: Simulated and experimental current single step responses plus reference.

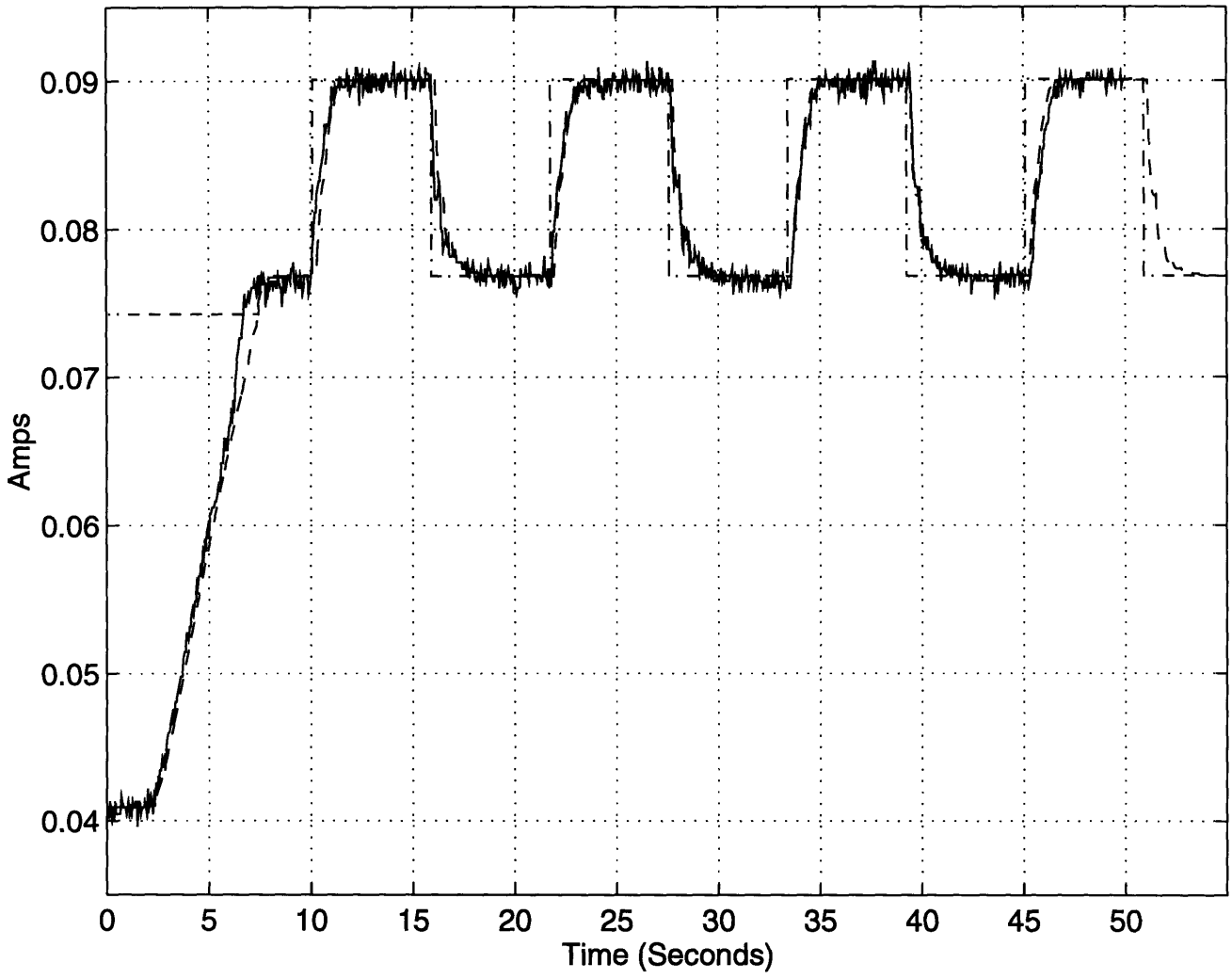

Figure 5-13: Simulated and experimental current oscillatory step responses plus reference.

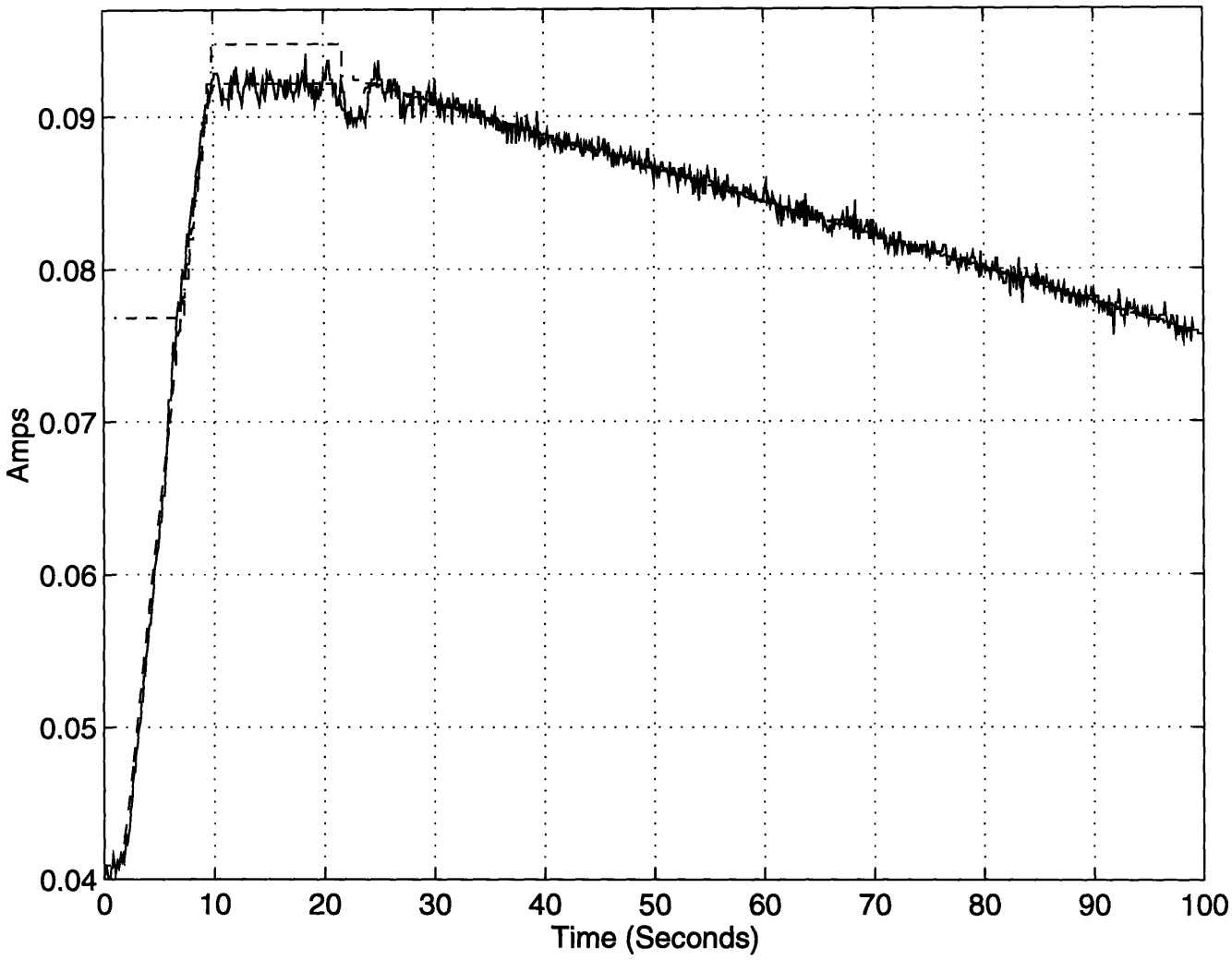

Figure 5-14: Simulated and experimental current ramp responses plus reference.

Effect of Steady State Band

Without steady state band control, quantization is expected to cause slight fluctuations in the amplitude of the input current for fixed output voltage. Quantization results in the voltage loop command k changing periodically even though the system has stabilized. An experiment using only the voltage loop evaluated steady state band control. Figure 5-15 shows the positive and negative peaks of many periods of input current for steady state band control disabled (solid line) and enabled (dashed line). The amplitudes of the solid line oscillate slightly, while the peaks of the dashed line remain more uniform. Another experiment compares 8 bits DAC resolution to 12 bit with no steady state band control. The peaks of the solid line in Fig. 5-16 are from 12 bit control, while the peaks of the dashed line are from 8 bit control. No great differences can be discerned, probably arising from only 10 bits resolution in the A/D.

5.3 Inductive Coupling System

The prototype for the inductive coupling system was operated with a $50kHz$ DC / AC conversion rate, a $300V$ DC input bus, and a 5.0Ω load. The turns ratio of the transformer was 250 :23, $L_\mu = 33.55mH$, and $L_{lk} = 5.26mH$. Evaluation of the inductive coupling prototype of the system in Figure 1-1 includes observing the signals related to the lossless switching scheme and analyzing the ability of the system to deliver power to a load.

5.3.1 Lossless Switching Signals

For convenience, Figure 3-7 is repeated in Fig. 5-17. The following examination of the voltages and currents during MOSFET switching transitions for the half-bridge DC / AC converter refers to the labels in the figure. Some of the figures in this section have unitless y-axes. These figures qualitatively compare the timing of various signals, and display waveforms that have been scaled to fit on one graph. In the captions below these figures, the waveforms will be named as they appear on the y-axis starting from the lower left hand corner and moving up the y-axis.

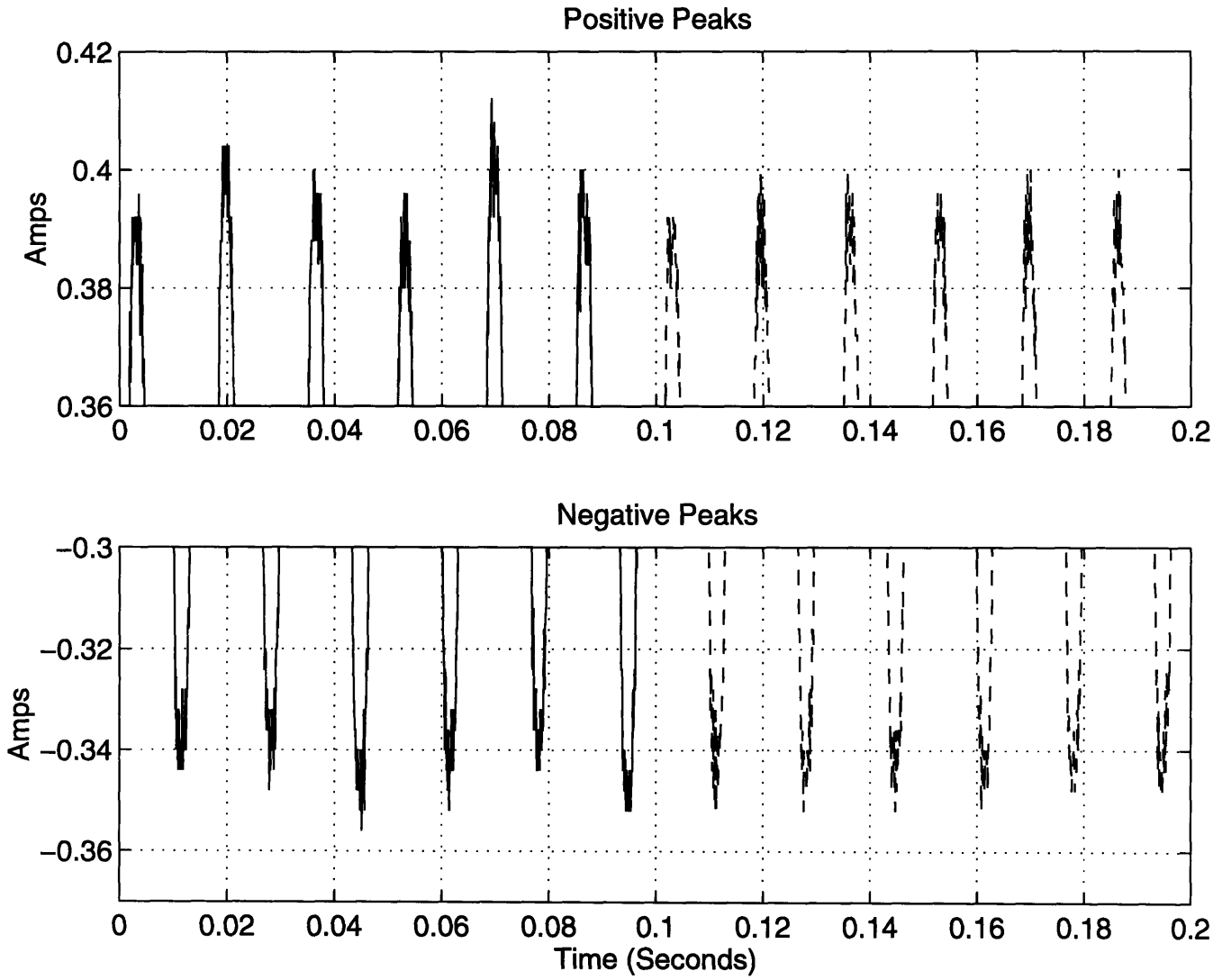

Figure 5-15: Comparison of input current with and without steady state band control.

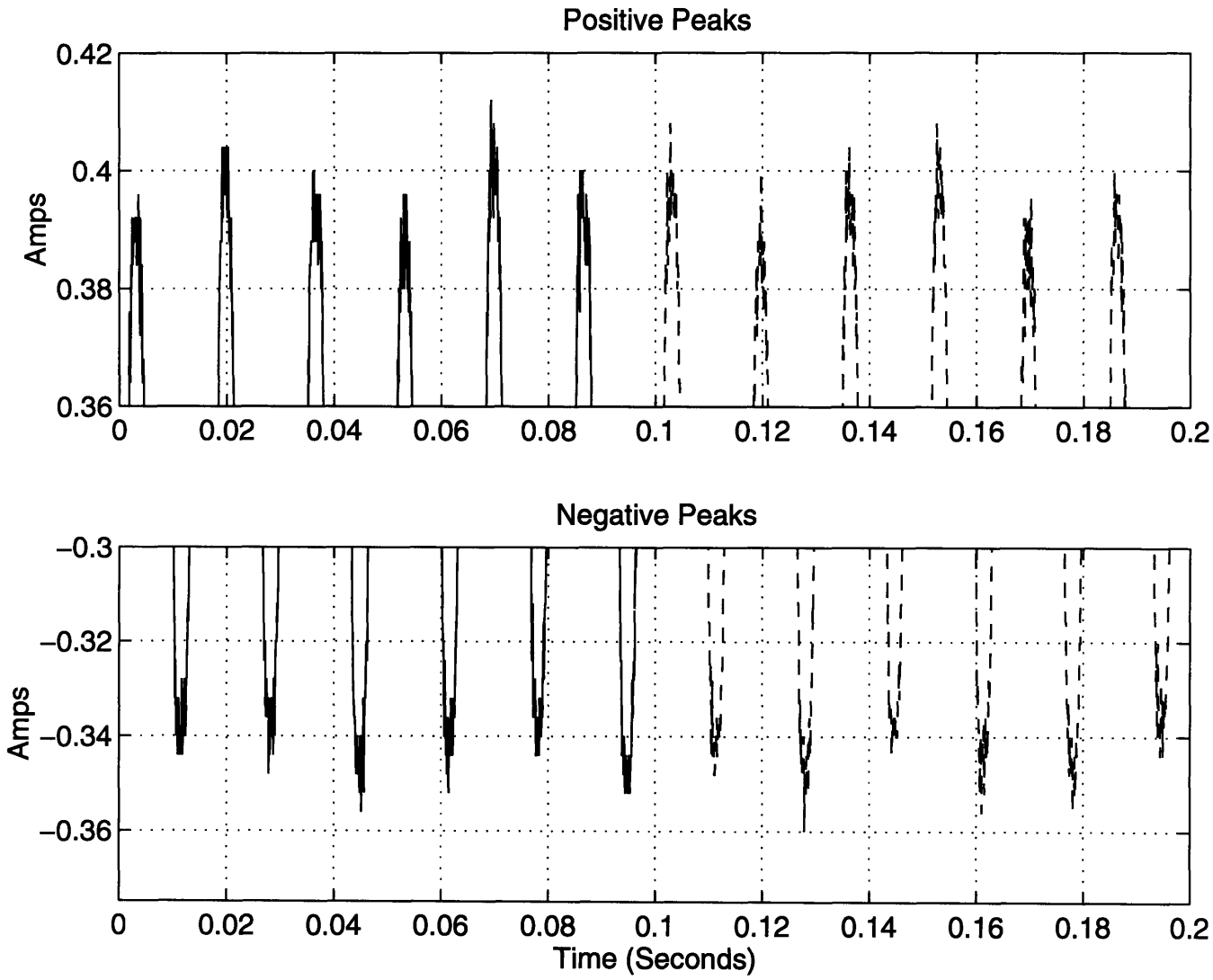

Figure 5-16: Comparison of input current: 12 bit DAC vs. 8 bit DAC resolution.

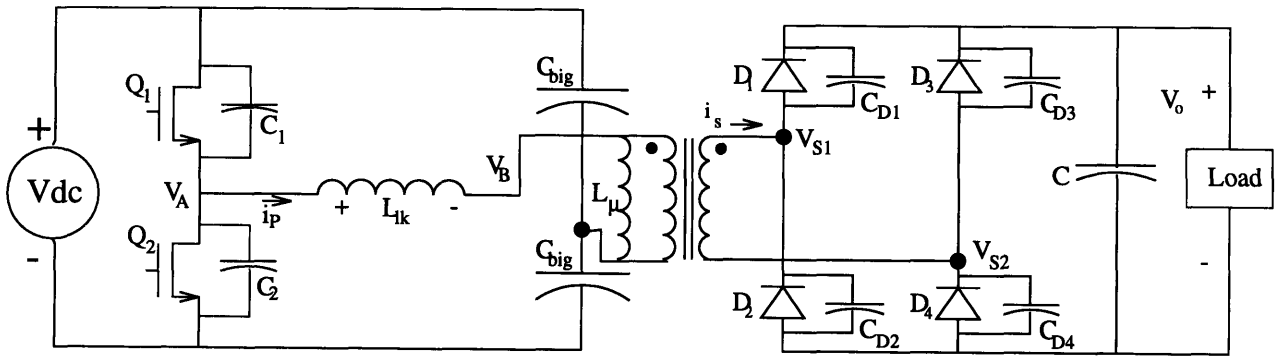

Figure 5-17: Inverter and rectifier with parasitic capacitances.

Gate Drive Signals

The high and low side MOSFET gate drive voltages (for $Q1$ and $Q2$, respectively) and high and low side digital switching signals are shown, from bottom to top, in Figure 5-18. The gate drive signals have been offset to be shown clearly in the figure, but they have not been scaled. These waveforms were obtained for a zero V_{DC} voltage bus at the input to the half-bridge inverter. Hence the floating gate drive for $Q1$ is referenced to 0 Volts. To account for the inversion in the DS0026 low side MOSFET gate driver, the low side digital signal in Fig. 5-18 is inverted. A slight, observable delay between the high side digital switching signal and the high side MOSFET gate drive indicates that the Unitrode 3724/3725 pair may not be an optimal circuit for high frequency DC / AC conversion.

Observation of Zero-Voltage Switching

From Figure 3-8, the voltage at node A is expected to ring to zero after $Q1$ turns off. The transistor $Q2$ can be subsequently turned on at any time with no volts across it. Additionally, the primary current i_p should ring while the voltage at node A rings, drop linearly while i_s is positive, ring when i_s reaches zero, then decline linearly after the diodes $D2$ and $D3$ conduct. Figure 5-19 demonstrates that these signals in fact behave according to Fig. 3-8. Figures 5-20 and 5-21 magnify in time the transition parts of the DC / AC period. All three figures are normalized in scaling, and show, from bottom to top, the transformer primary current, the voltage at node A, the low side digital switching signal, and the high side digital switching signal. In Fig. 5-20, the high side switching signal toggles off, V_A rings to zero, and i_p rings with V_A. When V_A reaches zero Volts, the linear decline of i_p occurs, during which time the low side switching signal toggles on to instigate zero-voltage $Q2$ turn-on. The coinciding behavior of i_s will be examined in a following section.

Comparison of Key Signals

The behavior of the secondary current in the scaled waveform Figures 5-22, 5-23, and 5-24 departs from Figure 3-10. All three figures show, from bottom to top, the secondary current, primary current, secondary voltage, and voltage at node A. Figures 5-23 and 5-24 magnify these waveforms in time. The discussion of i_s in Chapter 3 was inadequate to predict the behavior *actually* observed. Instead, the following development refers to Fig. 5-23, and theorizes about the new results.

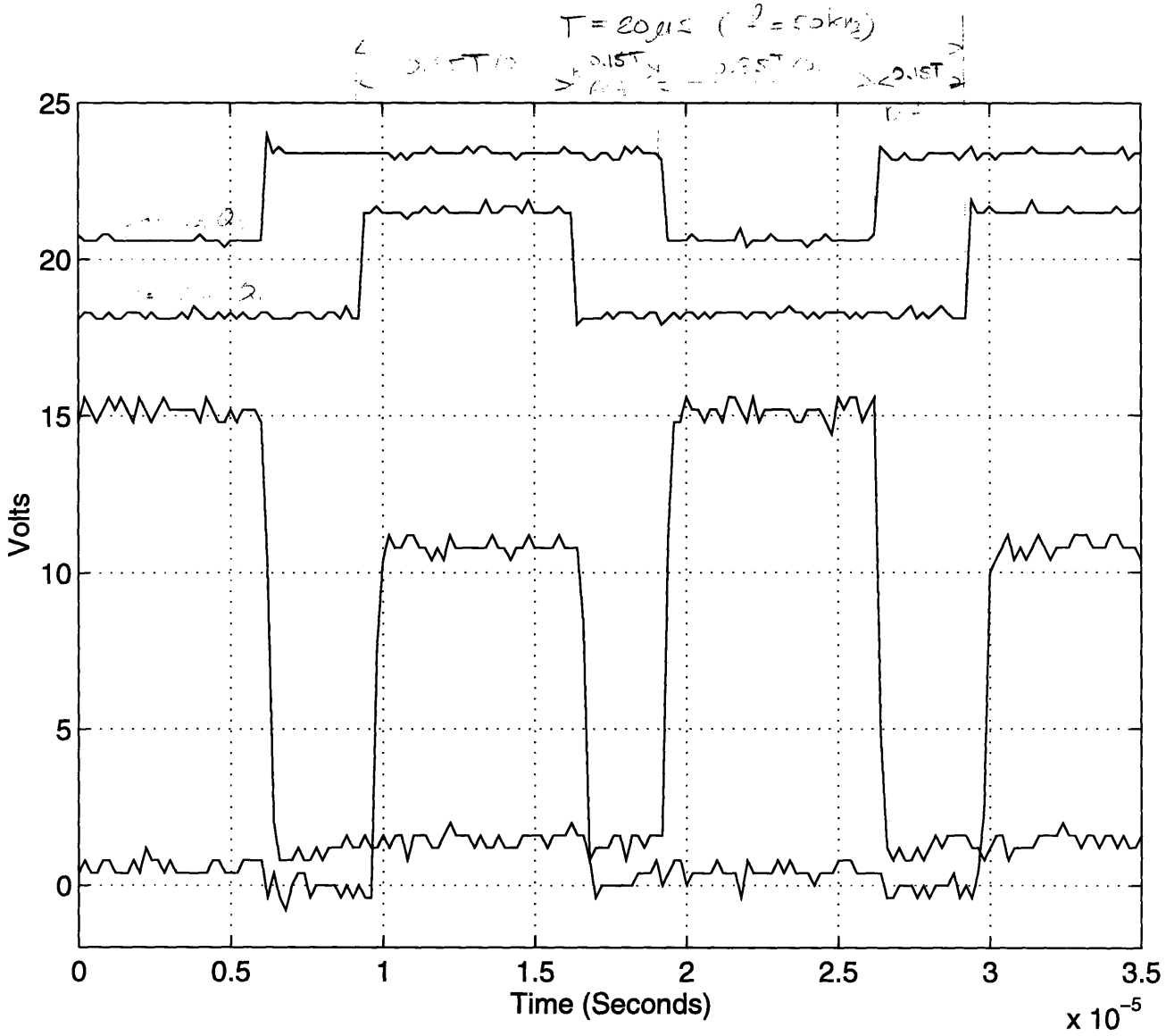

Figure 5-18: As they appear on y-axis, starting from bottom left corner and moving up y-axis: high side MOSFET gate drive, low side MOSFET gate drive, high side digital switching signal, low side digital switching signal.

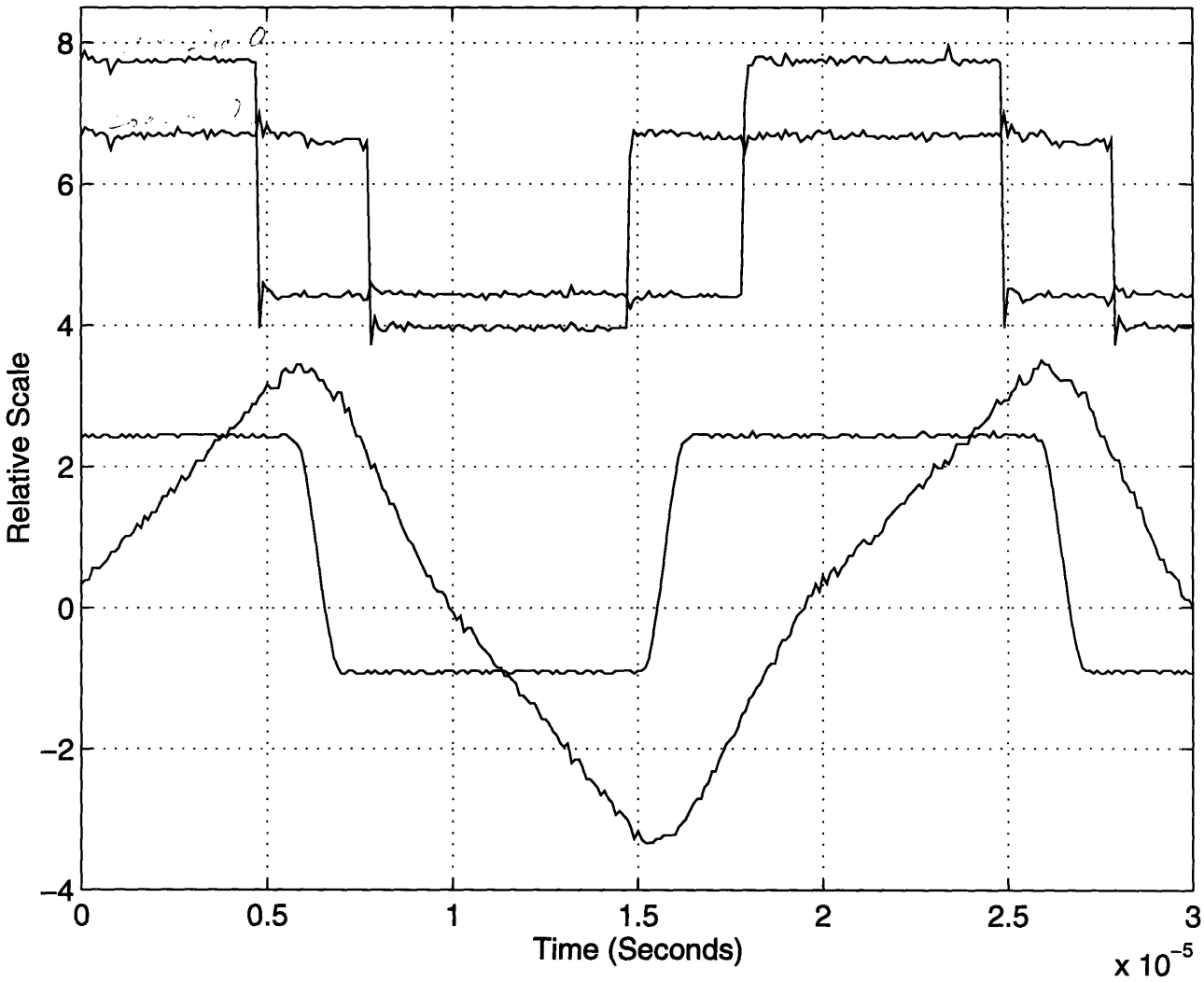

Figure 5-19: As they appear on y-axis, starting from bottom left corner and moving up y-axis: transformer primary current, voltage at node A, low side digital switching signal, high side digital switching signal

.

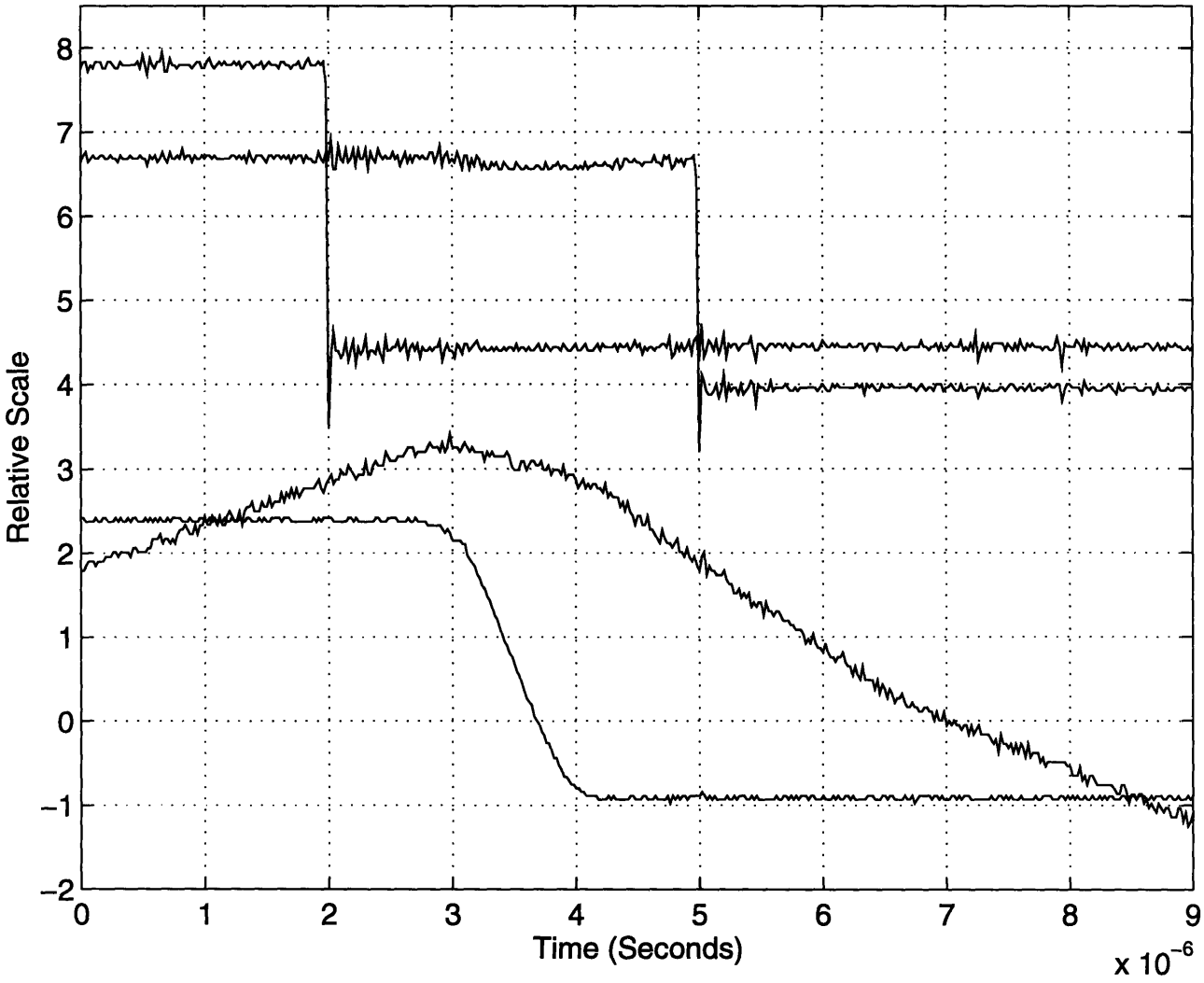

Figure 5-20: As they appear on y-axis, starting from bottom left corner and moving up y-axis: transformer primary current, voltage at node A, low side digital switching signal, high side digital switching signal

.

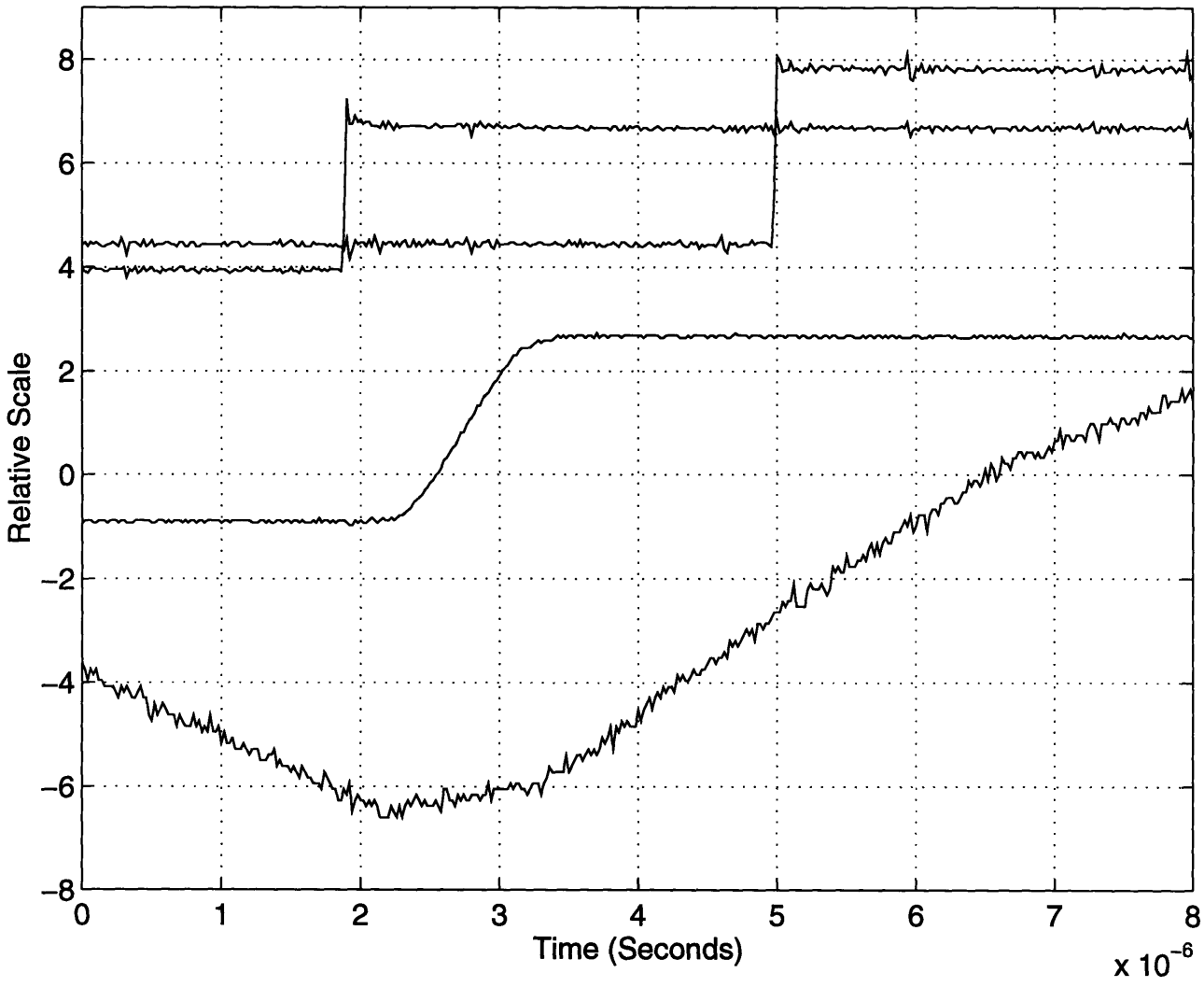

Figure 5-21: As they appear on y-axis, starting from bottom left corner and moving up y-axis: transformer primary current, voltage at node A, low side digital switching signal, high side digital switching signal

.

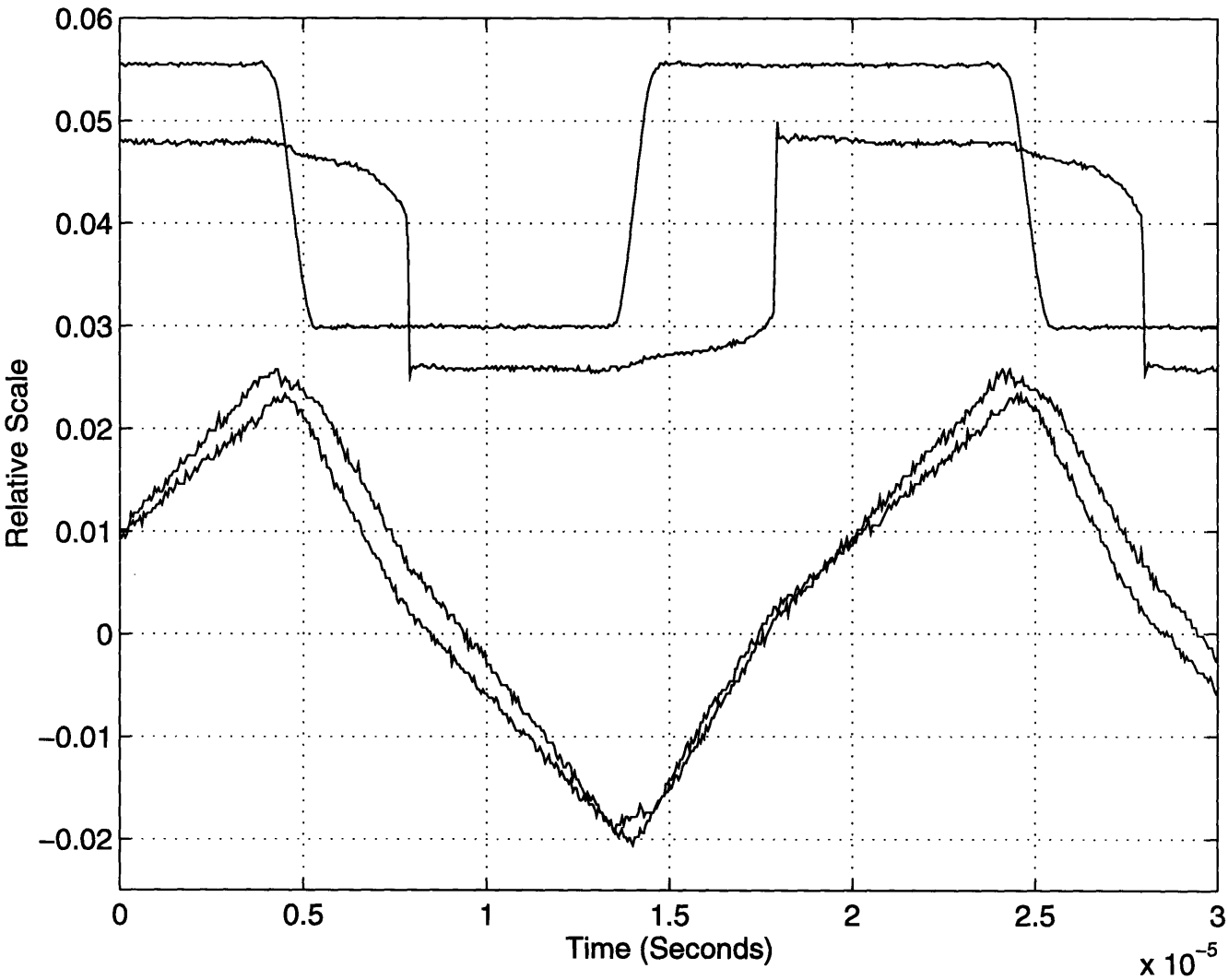

Figure 5-22: As they appear on y-axis, starting from bottom left corner and moving up y-axis: secondary current, primary current, secondary voltage, voltage at node A.

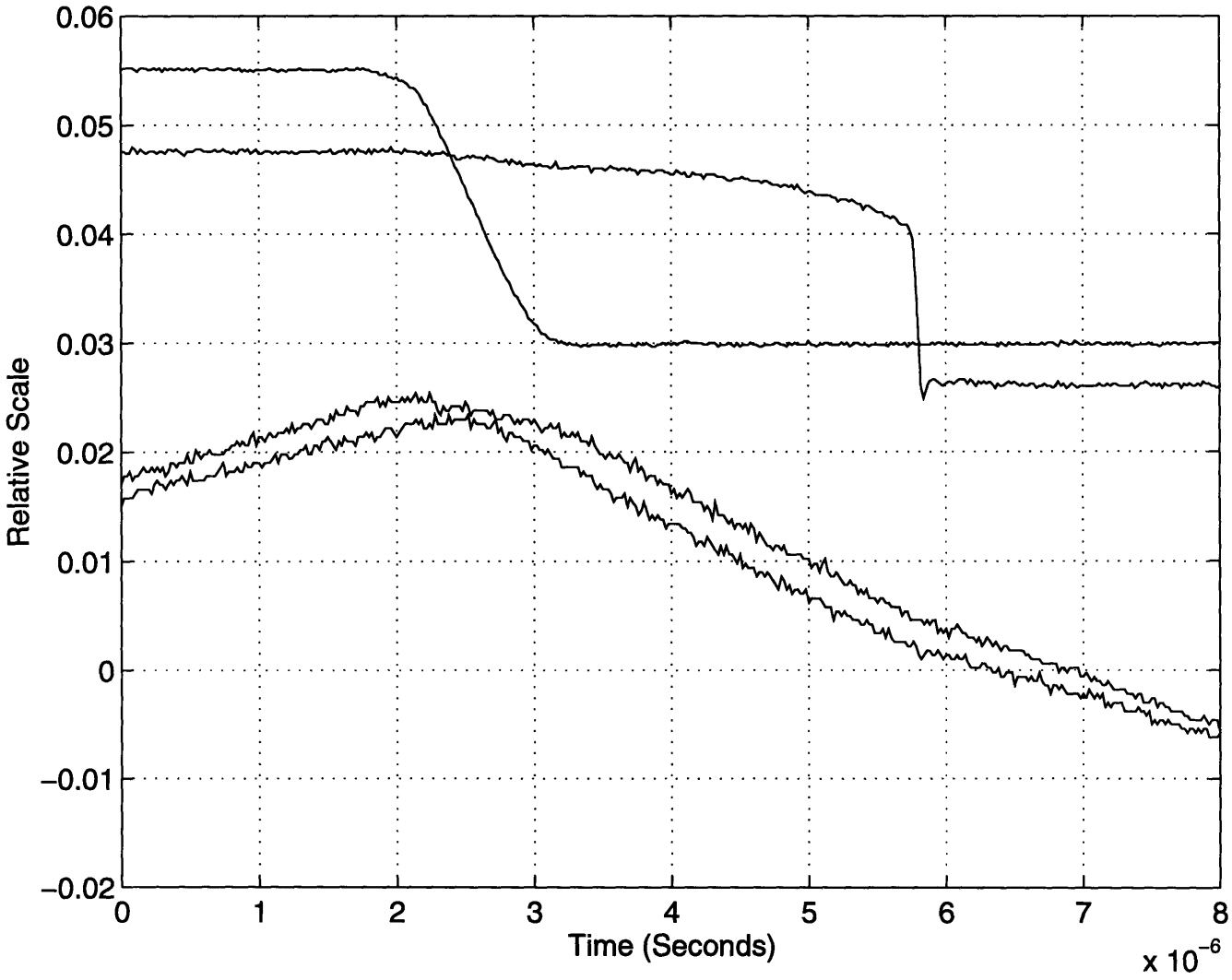

Figure 5-23: As they appear on y-axis, starting from bottom left corner and moving up y-axis: secondary current, primary current, secondary voltage, voltage at node A.

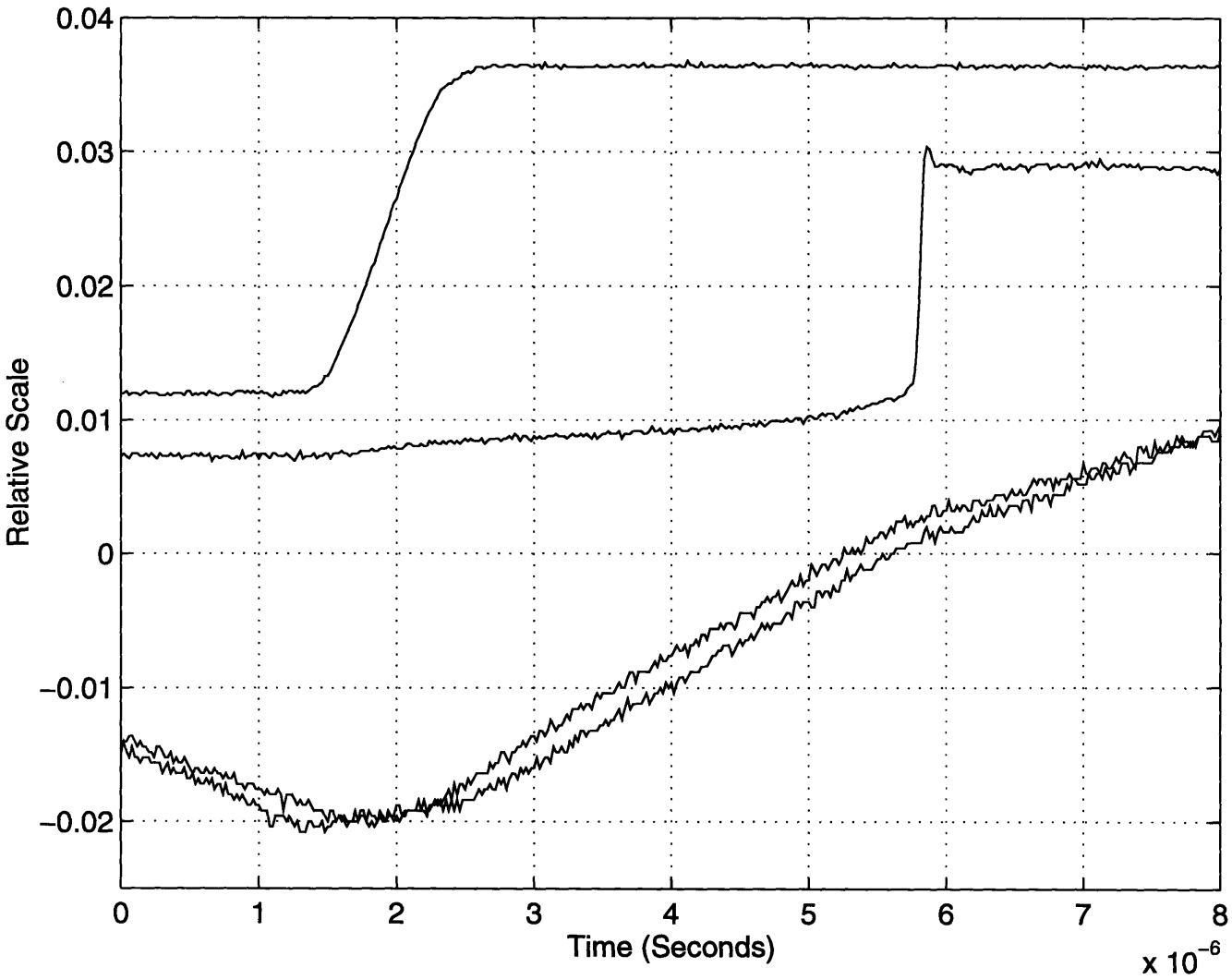

Figure 5-24: As they appear on y-axis, starting from bottom left corner and moving up y-axis: secondary current, primary current, secondary voltage, voltage at node A.

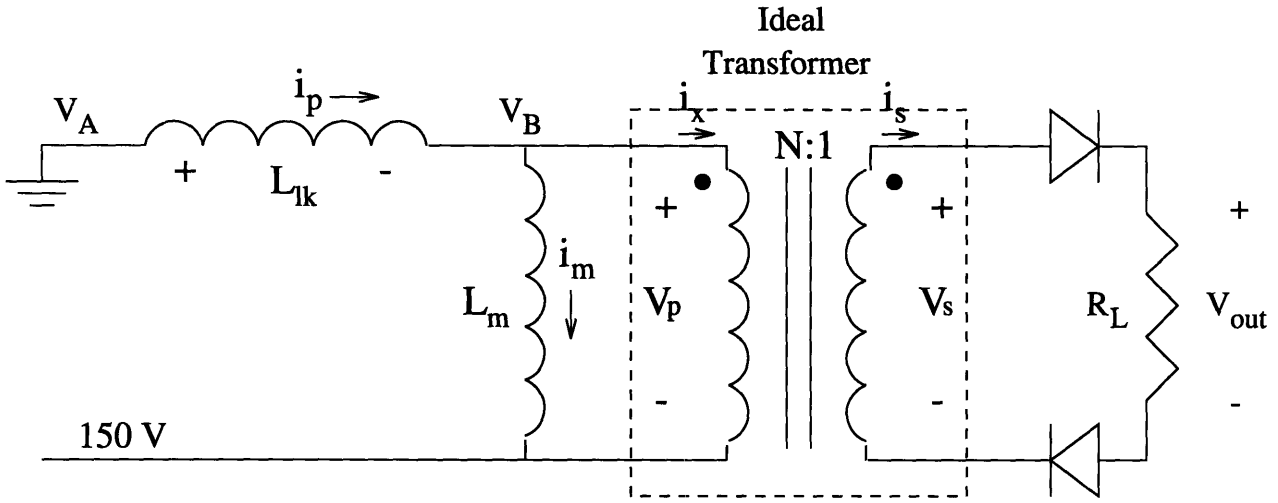

Figure 5-25: Equivalent circuit around the transformer during primary current linear decline.

While the i_p and i_s waveforms in the figure are scaled, their zero-crossings are exact in time. As expected, when V_A rings to zero, i_p rings. Thereafter, i_p falls linearly until i_s reaches zero. Also as expected, the voltage at the terminals of the transformer secondary switches polarity when the secondary current, i.e. the rectifier diode current, changes sign. This voltage sign reversal occurs because the load voltage V_o, which stays roughly constant, now is impressed on the secondary winding through diodes $D2$ and $D3$ rather than through $D1$ and $D4$.

The heretofore unexplained behavior lies in the slow decline of i_s during the transition, which is not shown in Figure 3-10, and the slight difference in the times at which i_p and i_s cross zero. Figure 5-25 simplifies the state of the DC / AC circuit during the time when i_p is falling linearly. Since i_s is still positive, then V_s is V_o plus two diode drops. The voltage primary voltage is $V_p = NV_s$. The node voltage V_B is equal to $150V$ (low side capacitor voltage) plus the ideal transformer primary voltage. Then the voltage across the leakage inductor is $-V_B$. From Table 3.1, for the values of load resistance and inductances given, the voltage $V_s = 4.56V$. Thus $V_p = 49.6V$ and $V_{lk} = -V_B = 199.6V$.

These voltages depart from Figure 3-8, which says that $V_{lk} = -300V$ and $V_p = 150V$ during this part of the switching transition. The reason for the difference stems from the new load regulatory effects summarized in Table 3.1 that are caused by the leakage inductance.

With these voltages, expected slopes of the currents i_p and i_s can be calculated using the constitutive law $\frac{V}{L} = \frac{di(t)}{dt}$. The slope of i_p is expected to be

$$\frac{V_{lk}}{L_{lk}} = \frac{-199.6V}{5.26mH} = -3.79 * 10^4 A/s. \tag{5.3}$$

From Kirchhoff's Current Law, the slope of i_s is calculated as the turns ratio N times the slope of i_p minus the slope of i_m, which is

$$N(\frac{V_{lk}}{L_{lk}} - \frac{V_p}{L_m}) = \frac{250}{23}(\frac{-199.6V}{5.26mH} - \frac{49.6}{33.55mH}) = -3.96 * 10^5 A/s. \tag{5.4}$$

From the graphs of i_p and i_s in Figure 5-27, the slopes in the experimental data can be approximated as $\frac{di_p(t)}{dt} = -3.4*10^4 A/s$, and $\frac{di_p(t)}{dt} = -3.5*10^5 A/s$. The experimental current slopes are 10.3% and 11.6% off the projected values, indicating substantial accordance given the inaccuracies inherent in the prototype circuit and transformer construction.

Figures 5-26 and 5-27 display, with units, the unscaled traces of V_A, the secondary voltage, i_p, and i_s over a period of operation of the half-bridge inverter.

5.3.2 Power to Load

Figure 5-28 displays the voltage V_{s1} with respect to the upper node of the load resistor in Fig. 5-17. When the diodes $D1$ and $D4$ are on, then V_{s1} should be equal to one diode drop. When the diodes $D2$ and $D3$ are on, then V_{s1} should be equal to minus the load voltage plus one diode drop (from Kirchhoff's Voltage Law).

Table 3.1 predicts that for the load resistor (5Ω) and the switching frequency ($50kHz$) in the prototype, the voltage at the terminals of the secondary should be $4.56V$. Then the voltage V_{s1} with respect to the load voltage should be $-4.56V + .9V = -3.66V$ while diodes $D2$ and $D3$ are on, and $.9V$ when $D1$ and $D4$ are on. Clearly, Figure 5-28 demonstrates this behavior of V_{s1}.

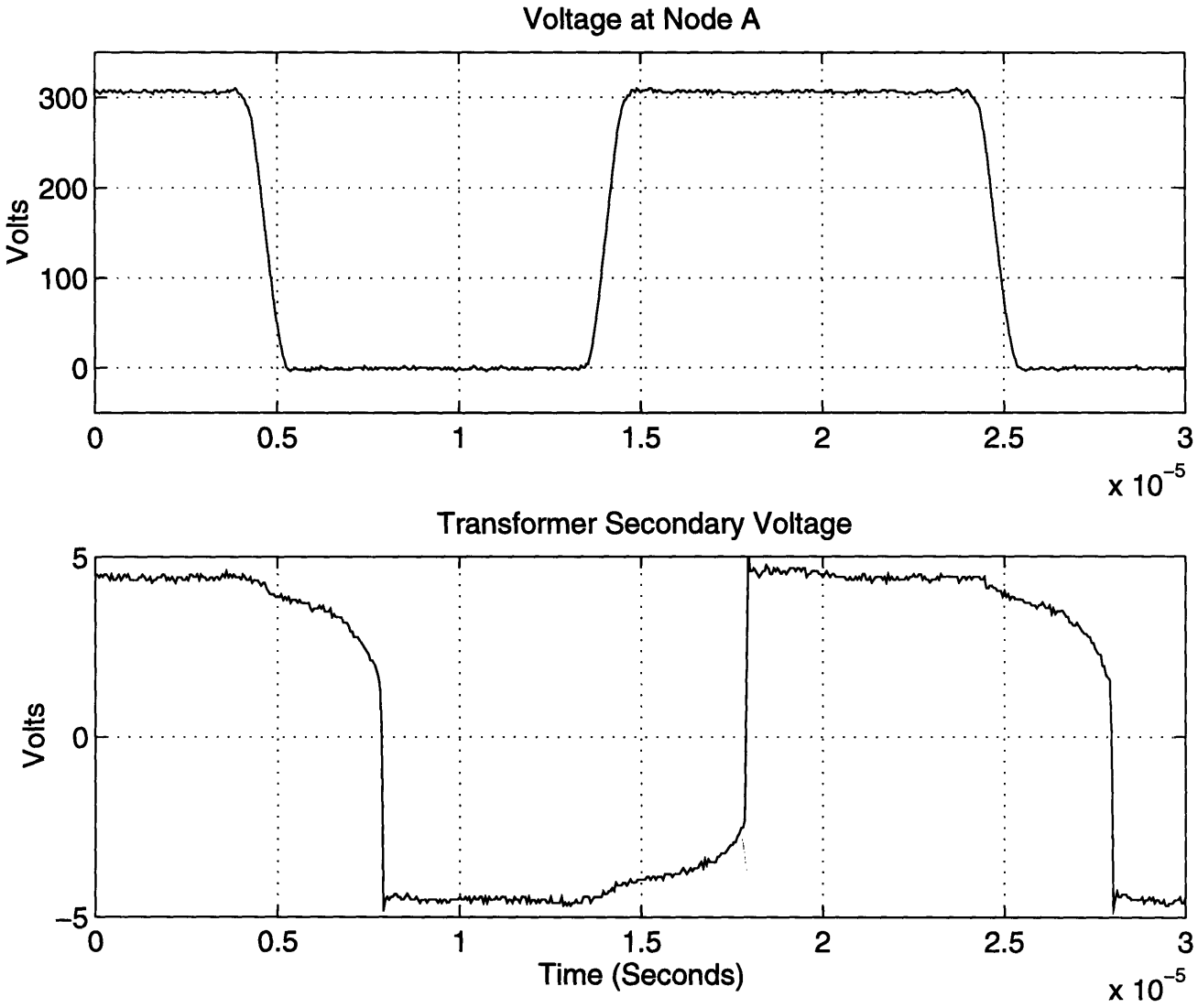

Figure 5-26: Transformer secondary voltage (bottom) and voltage at node A (top).

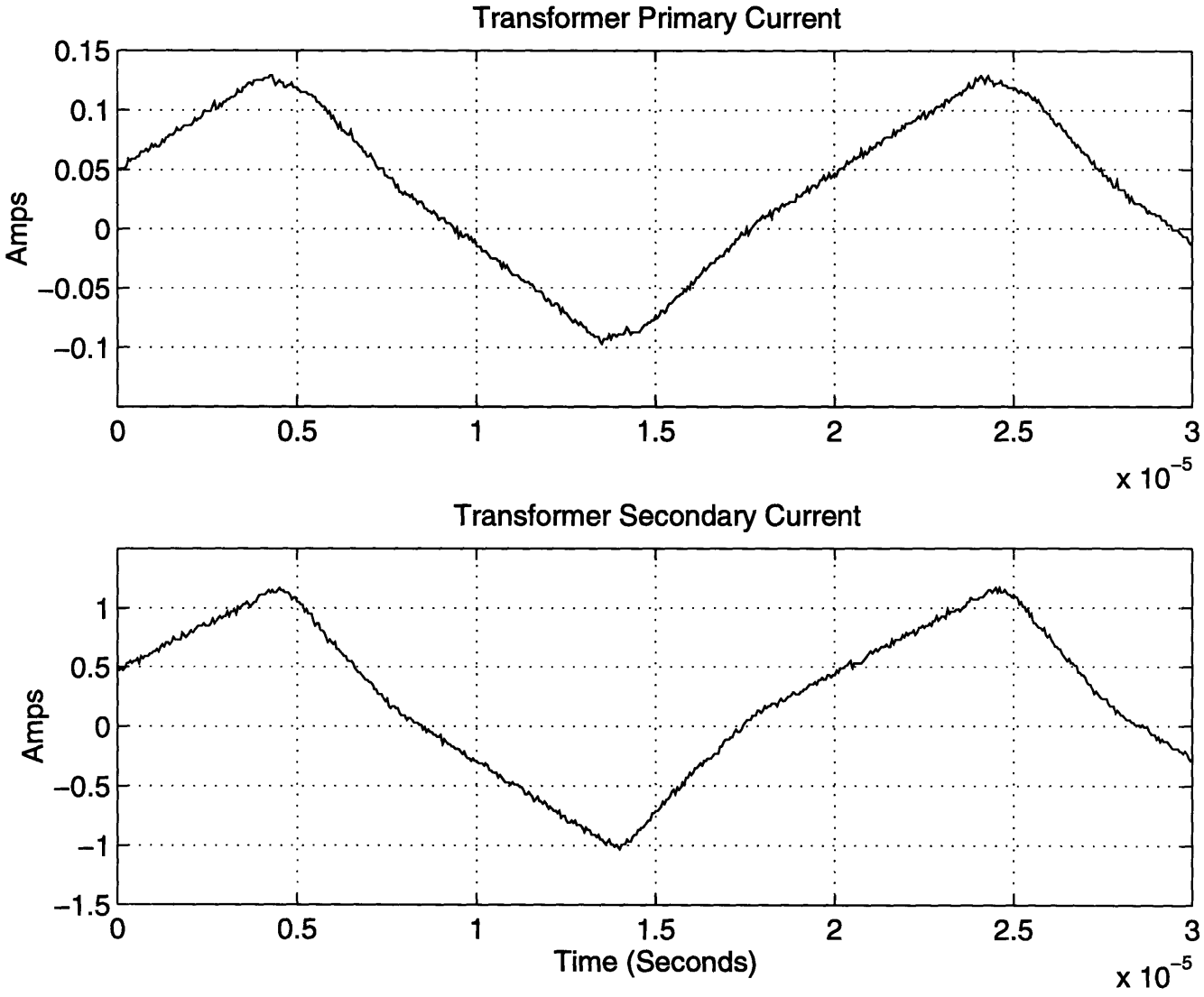

Figure 5-27: Transformer secondary current (bottom) and primary current (top).

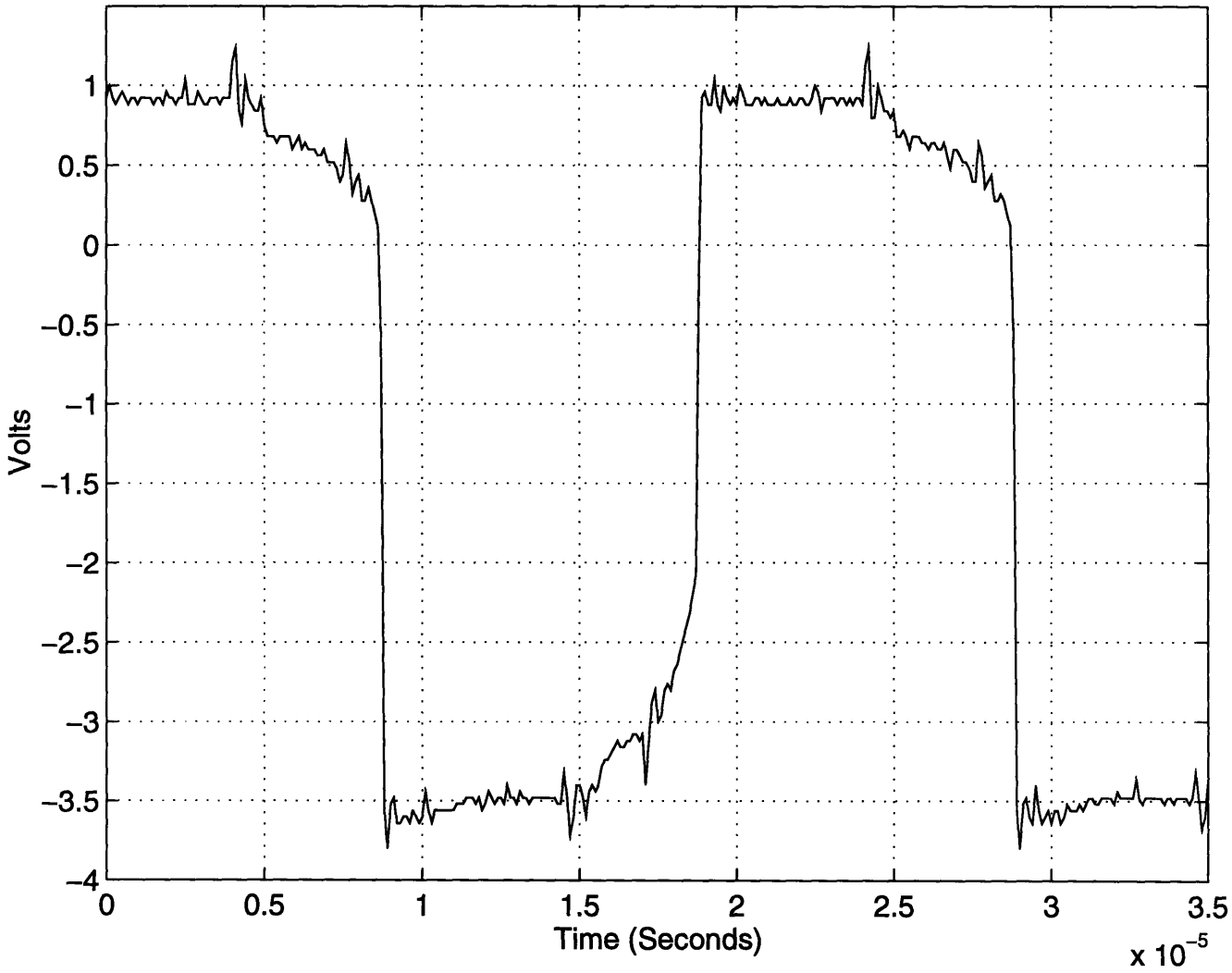

Figure 5-28: VS1 of the transformer secondary referenced to the load voltage.

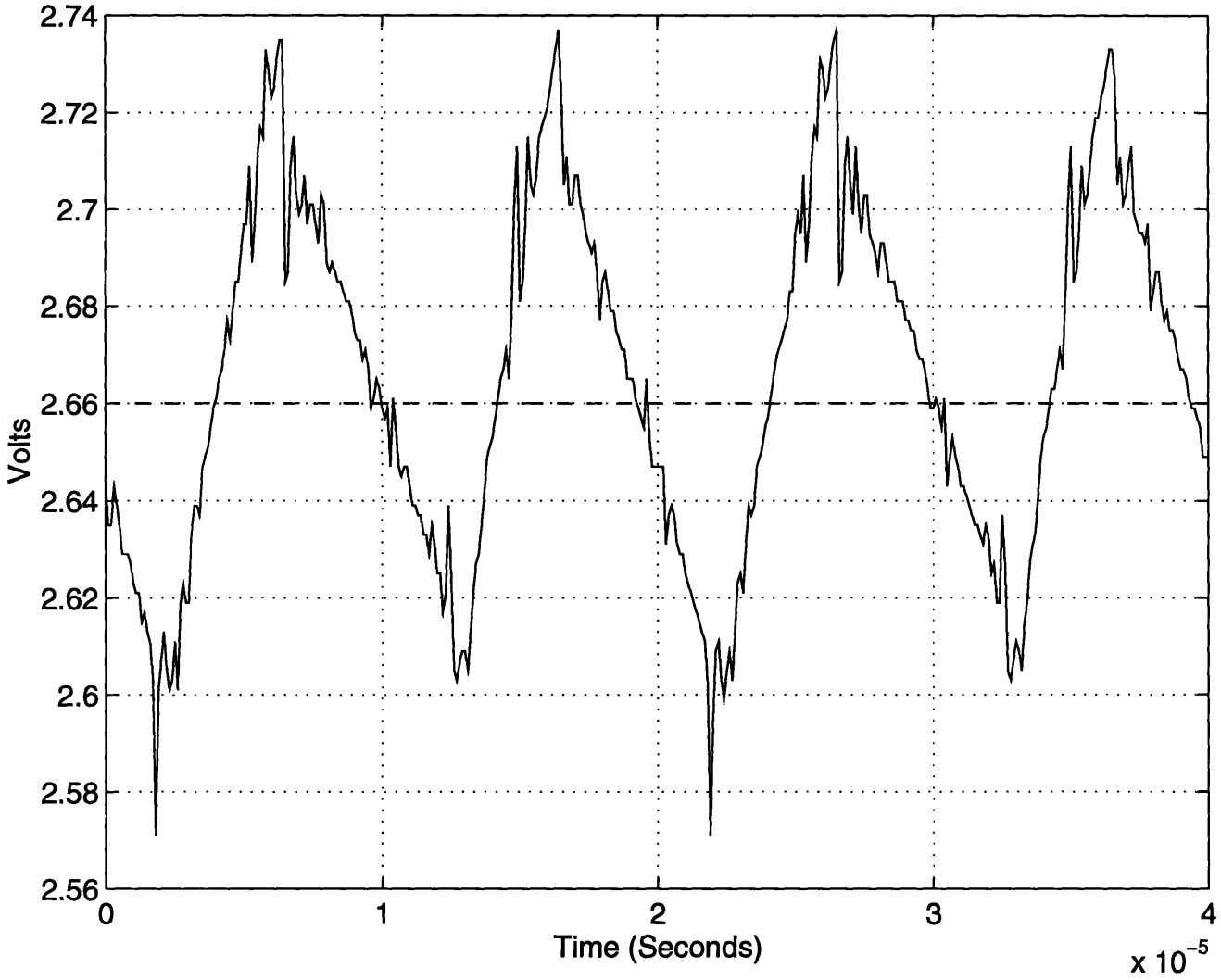

Figure 5-29: Load voltage, including ripple.

Additionally, the DC level of the load voltage was measured to be $2.66V$. As the load voltage should be equal to the peak value of the secondary voltage minus two diode drops, the value $2.66V$ almost exactly matches the expected load voltage $4.56V - (2 * .9V) = 2.76V$. Then the equations that lead to the values in table 3.1 are shown to be good models for the load regulation in the gapped core transformer topology of the circuit in Figure 5-17.

The small ripple in V_o, as shown in Figure 5-29, can be reduced by increasing the size of the rectifier capacitor, or by inserting a filtering inductor in series with the RC parallel network in the rectifier.

Chapter 6

Conclusions and Future Work

This project explored the design and implementation of an inductively coupled electric vehicle battery charging system. Goals included safety, reliability, and versatility. High efficiency was important with wide spread use of charging stations in mind. The prototype accomplishes these goals. The boost converter topology operates with unity power factor, facilitating maximum power transfer from the source. The model of the boost converter resulting from unity power factor leads to a robust, linear, large signal control scheme by which almost any battery with a linear model can be delivered an arbitrary charging current profile. The versatility afforded by the control scheme suggests a digital implementation whose flexibility enables easy changing of control code for different battery types. The successful magnetic coupling of energy from the charging station to the vehicle provides safety and reliability at the charging interface.

Starting from a successful unity power factor voltage loop control scheme developed by Mitwalli [34] [25, p. 396-9], an enhancement using feed forward eliminates voltage loop dependence on load. An outer current loop operates at a slower sampled data rate than the voltage loop, resulting in the simplified UPF plant delay model. With tools for developing sampled data equations for any linear load, the control scheme can be stabilized for linear controllers such as PI. Second order effects such as accumulator windup and quantization are combated through additional control features.

A half bridge inverter impresses an AC signal on the primary of a gapped core transformer. The transistors of the DC / AC circuit switch losslessly through a scheme from [35]. The

DC / AC inverter, transformer, and rectifier were built and tested. Experimental results prove that in fact the DC / AC conversion and rectification were occurring losslessly.

In order for the system to be feasible, three enhancements must be made in future work. First, the complete battery charging system of Figure 1-1 must be assembled, connected and tested since there was insufficient time during this project. The DC voltage at the half bridge inverter is the output of the boost converter. The load equations for the current loop controller will change since the battery will load the unity power factor DC supply through the DC/AC converter rather than directly. Feedback to the microcontroller is received through the capacitive coupling circuit from [43]. In short, the connection through an air gap of the UPF DC supply to the inductive and capacitive coupling systems as shown in Figure 1-1 must be made and tested.

The second improvement is to increase the amount of power that the charger can draw from the utility and efficiently supply to a battery. The prototype is designed to supply 250 Watts. Faster chargers will draw the maximum power possible from the utility, which for typical wall outlets is $2kW$. An important step towards a higher power prototype is reduction of the implementation to printed circuit boards. This improvement may provide reliability, better signal integrity, and safety to the circuit construction.

Third, the charger must take into account battery temperature and state of charge.

Another improvement may lie in developing a more sophisticated DC / AC conversion. Dividing the power to be transferred from the DC source to the battery among several bridges will reduce the size of each transformer due to smaller power demands in each branch, and increase reliability by lessening the loss when one branch fails. A "polyphase-shifted PWM" lossless switching scheme for the branch phases would need to be derived.

For commercial proliferation of this technology, the charging station will need some method for simple recognition of the electric vehicle battery type. The charging station might have a bank of controllers for the different battery loads. Alternatively, manufacturers might store charging information in the vehicle to be transmitted over the transceiver interface. The program running at the charging station will require knowledge of the optimal charging profile for the given battery. A graphical user interface for either system may be useful.

Appendix A

Accumulator Maximum Calculations

This appendix derives the equations for the voltage and current loop digital accumulator maximum values. In both cases, the maximum accumulator value is taken to be that number which, when multiplied by the relevant digital PI gain while the PI error term is zero, results in the maximum allowable command signal.

A.1 Voltage Loop Accumulator

The calculation for σ_{vdmax} arises from the digital voltage loop PI equation

$$k_d[n] = \frac{1}{V_d^2 G_{dig}}(h_{1d}(V_{od}^2 - v_{od}^2) + h_{2d}\sigma_{vd}) + h_{pd}\frac{v_{od}i_{od}}{V_d^2}. \tag{A.1}$$

If the maximum k_d is 4095, and

$$V_{id} = F_{AD}\, D_{ivi}\, V, \tag{A.2}$$

then assuming zero steady state voltage error, the equation for σ_{vdmax} is

$$\sigma_{vdmax} = \frac{4095 V_{id}^2 G_{dig}}{h_{2d}}. \tag{A.3}$$

For $V_{id} = 7.6 * 10^{-3} * 204.6 * 168$, $G_{fig} = .5$, and $h_{2d} = 973/256$, then $\sigma_{vdmax} = 36762754$.

A.2 Current Loop Accumulator

Assuming a maximum output voltage, then the scaled version is

$$v_{odmax} = m\, F_{AD}\, D_{ivo}\, v_{omax}. \tag{A.4}$$

Assuming no steady state current error, then from the PI equation

$$V_{od}[N] = h_{3d}(I_{od} - i_{od}) + h_{4d}\sigma_{id}, \tag{A.5}$$

then

$$\sigma_{idmax} = \frac{V_{odmax}}{h_{4d}}. \tag{A.6}$$

For $V_{odmax} = 4.19 * 9.75 * 10^{-3} * 204.6 * 400$ and $h_{4d} = 4497/1024$, then $\sigma_{idmax} = 761$.

Appendix B

Passive Filters for Anti-Aliasing

This appendix presents the anti-aliasing filters used to filter the boost converter output voltage and output current.

B.1 Output Voltage

The boost converter output voltage may contain a small $120Hz$ ripple from the rectified input voltage excitation. The control calculations sample at $120Hz$. To satisfy the Nyquist sampling criterion and eliminate aliasing, a low pass filter removes the $120Hz$ component from the output voltage.

For convenience, the passive, second order low pass filter is shown in Fig. B-1. The low pass filter contains three such low pass stages with buffering as shown in Fig. B-2. Three op amp buffering stages shield the input and the $H(S)$ stages from each other.

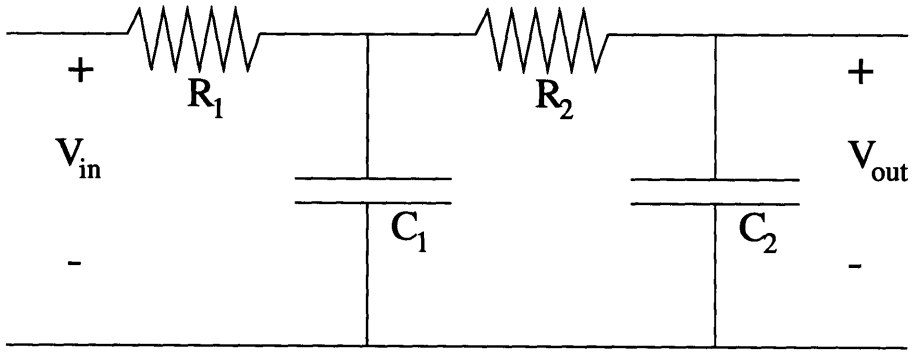

Figure B-1: Low pass filter stage.

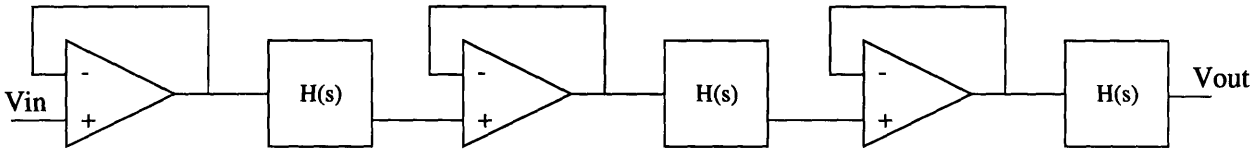

Figure B-2: Cascade of three filters with buffering.

Part	Value
R_1	280Ω
R_2	$2.25k\Omega$
C_1	$1\mu F$
C_2	$3.3\mu F$

Table B.1: Low pass filter part values.

The transfer function of the circuit is

$$H(S) = \frac{1}{R_1R_2C_1C_2s^2 + (R_1C_2 + R_1C_1 + R_2C_2)s + 1}. \quad \text{(B.1)}$$

For all three stages the values are shown in Table B.1.

Figure B-3 shows the Bode plot for one stage. The connected system's frequency response is Fig. B-4. At $2\pi * 120Hz = 754rad/sec$, the magnitude response is -56dB.

B.2 Output Current

A two stage low pass filter is used at the output of the current sensing circuit. The values in both stages are given in Table B.2. The Bode plot in Figure B-5 shows the two stage filter frequency response.

Part	Value
R_1	$18k\Omega$
R_2	$18k\Omega$
C_1	$560nF$
C_2	$560nF$

Table B.2: Low pass filter part values.

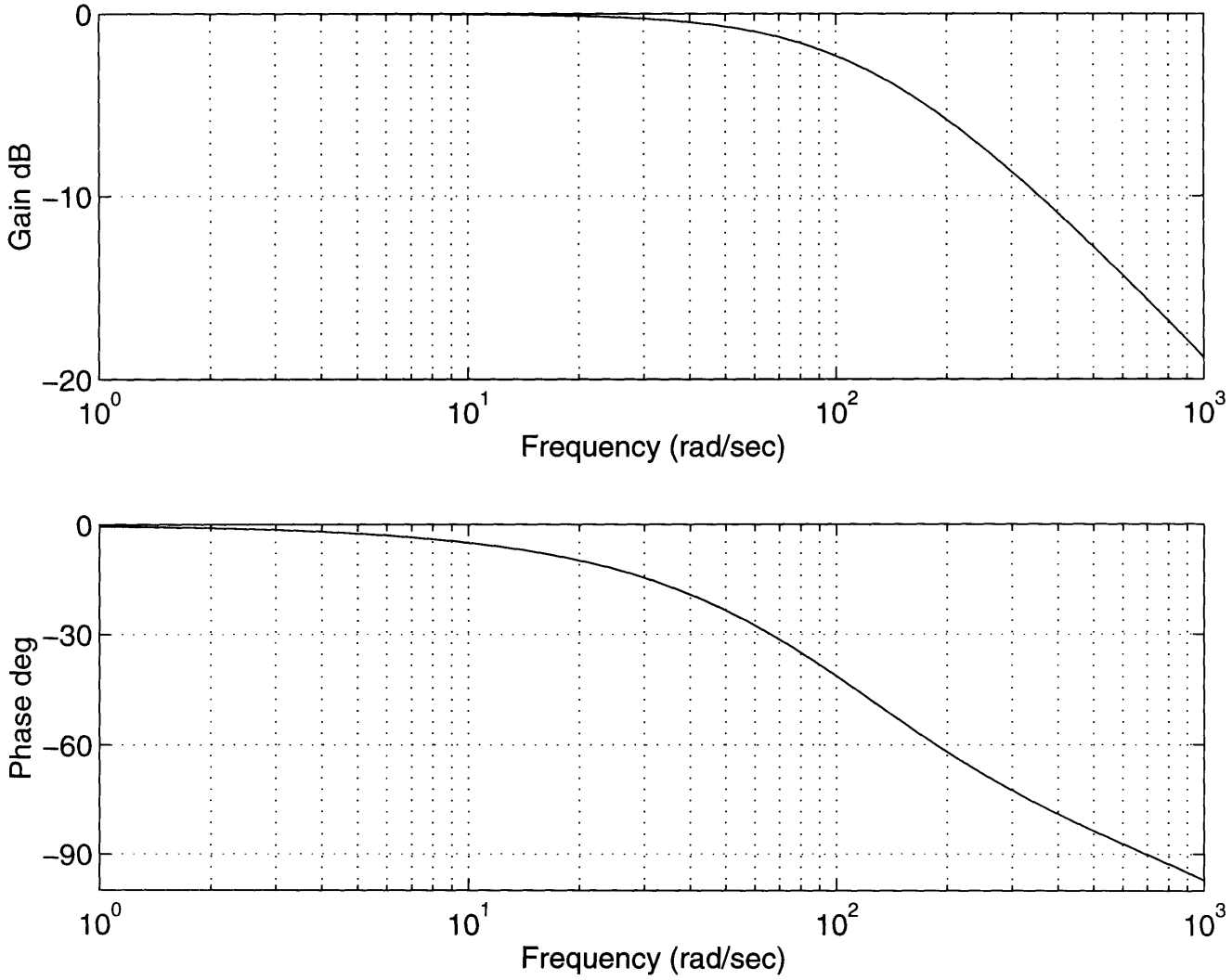

Figure B-3: Bode plot for one RC-RC stage of the output voltage filter.

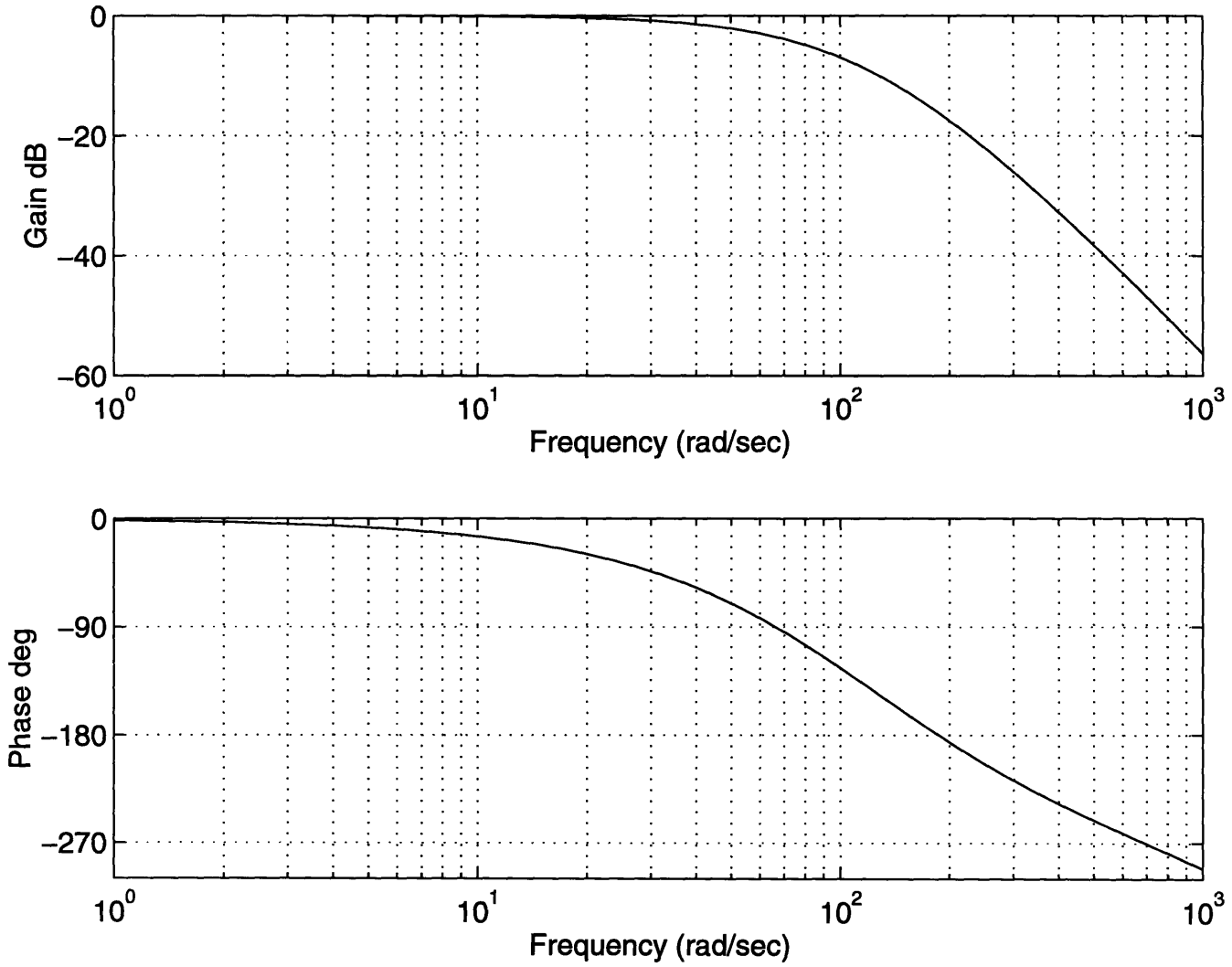

Figure B-4: Bode plot for three cascaded RC-RC stages of the output voltage filter.

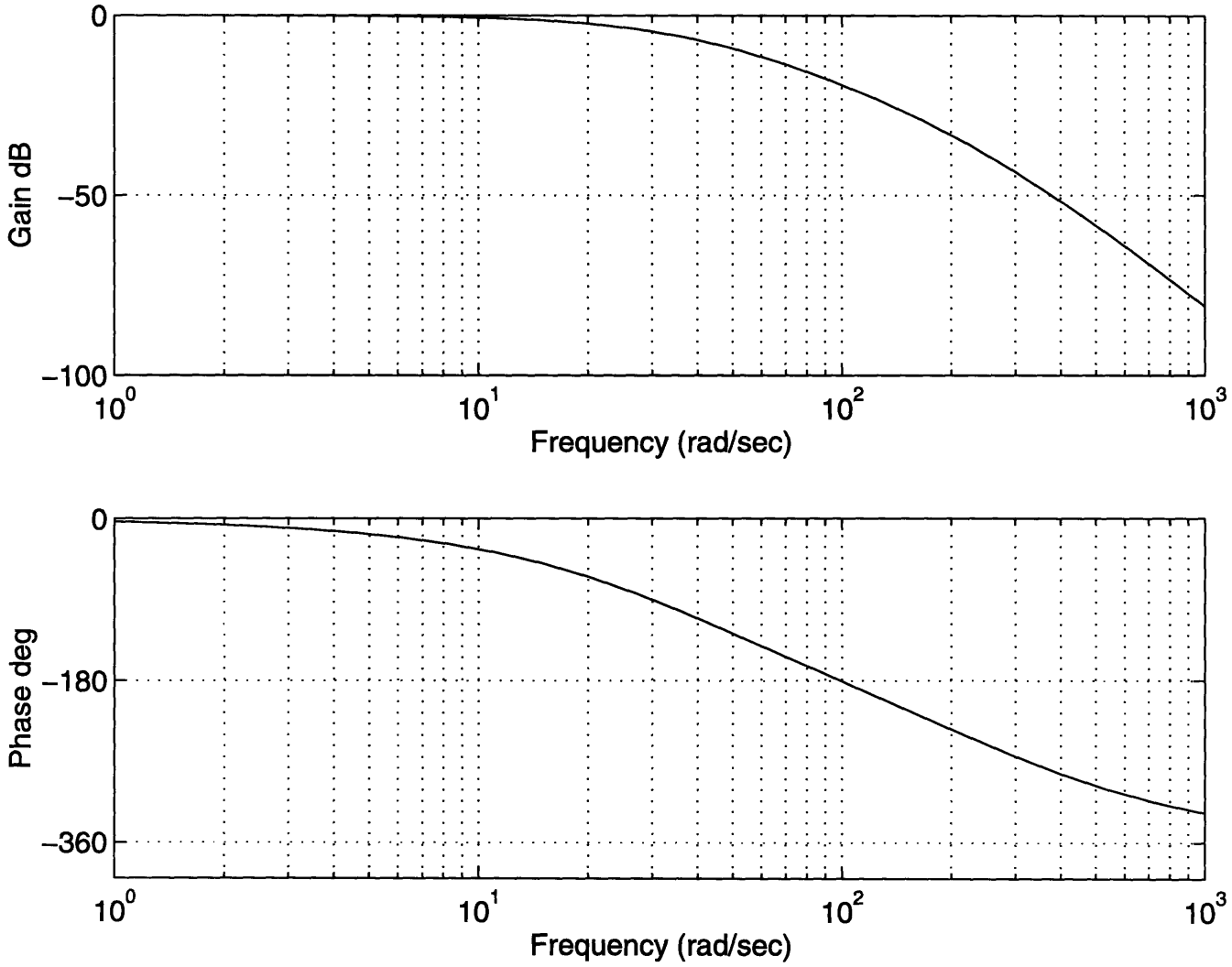

Figure B-5: Bode plot for two cascaded RC-RC stages of the output current filter.

Appendix C

Manual

The following procedures outline actions to perform in actually running the system. These routines include setting the maximum power levels, executing the code, activating the charging current control system hardware, and uploading pictures from the oscilloscope.

C.1 Maximum Power Level

The inputs A, B, and C to the UC3854 define the current reference by $\frac{AB}{C}$. Using the multiplying DAC, the input B is a scaled version of the rectified input voltage. With C fixed, altering A will change the maximum controllable current, since the input B is limited by the multiplying DAC. Limiting input current is equivalent to limiting input power, since the input voltage is always the utility AC voltage. For a resistive boost converter load, a limitation in input power results in an output voltage maximum. It is important to take care while raising the max input power so as not to exceed the voltage and energy ratings of parts such as the boost converter bus capacitor.

C.2 Executing the Code

To compile a program, the header file 80C196.H needs to be included in the code. Additionally, the processor's memory mapping requires the two lines
register char apple[9];
#pragma locate (apple = 0x30)
to be at the beginning of the code for proper operation. To run the monitor, simply type
ECM96
The EV80C196KB Embedded Controller Monitor allows downloading of the resulting .OUT file. Fortunately, the linking produces already located code. Therefore, after typing
LOAD FILENAME.OUT
only typing GO at the prompt is necessary. Displaying memory can be done with the commands BYTE, WORD, and DWORD for 1, 2, and 4 byte accesses, respectively. Changing the value in register memory is done by entering a new value at the prompt.

C.3 Activating the Hardware

There are two modes in which the powered boost converter works. If the Unitrode controller is not powered, then no power factor correction will occur, and the boost converter output will be $160V$. When the UC3854 is powered, the input current will match the input voltage in phase and wave shape based on the three inputs to the UC3854 controller. The safest procedure for activating the system is as follows.

- Turn on the $+15/-15V$ supply to ensure the inputs to the UC3854 are ready.
- Make sure the UC3854 power supply is off.
- Turn the knob on the VARIAC to "zero."
- Turn on the utility switch, allowing current to flow into the VARIAC primary.
- Slowly increase the VARIAC secondary voltage until the RMS value is $120V$.

- Download and execute a body of code writing zero to the XDAC. When the UC3854 is powered, then there will be no sudden transients since despite power factor correction, the inner current loop will command no extra power.
- Turn on the UC3854 power supply.
- Download and execute the application of the day.

To deactivate the system,

- Turn off the UC3854 power supply.
- Allow enough time for the utility input current to return to the uncorrected rectifier current wave shape.
- Turn off the utility input switch.
- Decrease the VARIAC knob to "zero" to prevent future non-zero accidental turn-on.

C.4 Oscilloscope Pictures

A serial cable connects a PC and a Tektronix TDS 320 oscilloscope. To run the file GETWFM.BAS on a PC, Quick Basic must first be invoked by typing QB. Executing OPEN in the FILE pull-down menu will load in GETWFM.BAS. The oscilloscope features two "REF" storage banks. To choose between them, change the relevant text in GETWFM.BAS to "REF1" or "REF2." The file name WFMXXXX.PRN to write to must also be changed in the file. START from the RUN menu will execute the upload procedure. The program CHANGEFILES.C will create a WFMXXXX.MAT MATLAB loadable file from the WFMXXXX.PRN wave file. The programs GETWFM.BAS and CHANGEFILES.C are contained in Appendix E.

Appendix D

Schematics

The following tables list all the parts in the prototype hardware. The subsequent schematics detail the prototype circuitry.

Parameter	Component	Number
D1	DIODE	MUR1560
D2	DIODE	MUR1560
D3	DIODE	MUR1560
D4	DIODE	MUR1560
D5	DIODE	MUR1560
D6	RECT	MDA970G6
D7	RECT	MDA970G6
Q1	MOSFET-N	IRFP450
Q2	MOSFET-N	IRFP450
Q3	MOSFET-N	IRFP450

Table D.1: Table of diodes and transistors.

Parameter	Value	Units
T1	XFMR1	-
T2a	XFMR2a	-
T2b	XFMR2b	-
T3	XFMR3	-
IND1	1.025	mH

Table D.2: Table of magnetic parts.

Parameter	Value	Units
R1	10	Ω
R2	10	$k\Omega$
R3	4.7	$k\Omega$
R4	2.2	$k\Omega$
R5	7.5	$k\Omega$
R6	10	Ω
R7	5.0	Ω
R8	30	$k\Omega$
R9	15	$k\Omega$
R10	10	$k\Omega$
R11	TRIMPOT,10	$k\Omega$
R12	10	$k\Omega$
R13	10	$k\Omega$
R14	10	$k\Omega$
R15	10	$k\Omega$
R16	20	$k\Omega$
R17	18	$k\Omega$
R18	18	$k\Omega$
R19	18	$k\Omega$
R20	18	$k\Omega$
R21	240	Ω
R22	382	Ω
R23	2.17	$k\Omega$
R24	5.1	$k\Omega$
R25	TRIMPOT,50	$k\Omega$
R26	356	$k\Omega$

Table D.3: Table of resistor values.

Parameter	Component	Number
R27	3.59	$k\Omega$
R28	20	$k\Omega$
R29	20	$k\Omega$
R30	4.75	$k\Omega$
R31	4.75	$k\Omega$
R32	287	Ω
R33	2.21	$k\Omega$
R34	287	Ω
R35	2.21	$k\Omega$
R36	287	Ω
R37	2.21	$k\Omega$
R38	TRIMPOT,200	$k\Omega$
R39	2.17	$k\Omega$
R40	5.1	$k\Omega$
R41	62	$k\Omega$
R42	82	$k\Omega$
R43	135	$k\Omega$
R44	2.17	$k\Omega$
R45	5.1	$k\Omega$
R46	6.6	Ω
R47	.235	Ω
R48	15	$k\Omega$
R49	22	$k\Omega$
R50	20	Ω
R51	14.34	$k\Omega$
R52	10	$k\Omega$
R53	1.6	$k\Omega$
R54	4.41	$k\Omega$
R55	3.91	$k\Omega$
R56	3.91	$k\Omega$
R57	24	$k\Omega$
R58	TRIMPOT,20	$k\Omega$
R59	TRIMPOT,20	$k\Omega$
R60	TRIMPOT,5	$k\Omega$
R61	3.91	$k\Omega$

Table D.4: Table of resistor values (cont'd).

Parameter	Value	Units
C1	1	μF
C2	1	μF
C3	680	pF
C4	13.3	μF
C5	470	μF
C6	470	μF
C7	1	mF
C8	560	nF
C9	560	nF
C10	560	nF
C11	560	nF
C12	.1	μF
C13	1	μF
C14	10	nF
C15	5.6	nF
C16	1	μF
C17	3.3	μF
C18	1	μF
C19	3.3	μF
C20	1	μF
C21	3.3	μF
C22	10	nF
C23	5.6	nF
C24	.1	μF
C25	.47	μF
C26	10	nF
C27	5.6	nF
C28	.47	μF
C29	470	μF
C30	1	μF
C31	100	μF
C32	1	nF
C33	.15	μF
C34	270	pF
C35	620	pF
C36	62	pF
C37	1	μF
C38	2500	μF

Table D.5: Table of capacitor values.

Parameter	Chip
U1	DS0026
U2	UC3724
U3	UC3725
U4	74LS163
U5	74LS163
U6	27C256
U7	74LS175
U8	LM358
U9	LM358
U10	LM317T
U11	LEM
U12	LM555
U13	AD204
U14	LM358
U15	LM358
U16	LM324
U17	LM555
U18	AD204
U19	LM358
U20	MAX501
U21	LM555
U22	AD204
U23	UC3854
U24	LM358

Table D.6: Table integrated circuits.

Parameter	Part
F1	FUSE 3A
F2	FUSE 2A
X1	XTAL 10MHz

Table D.7: Table of other parts.

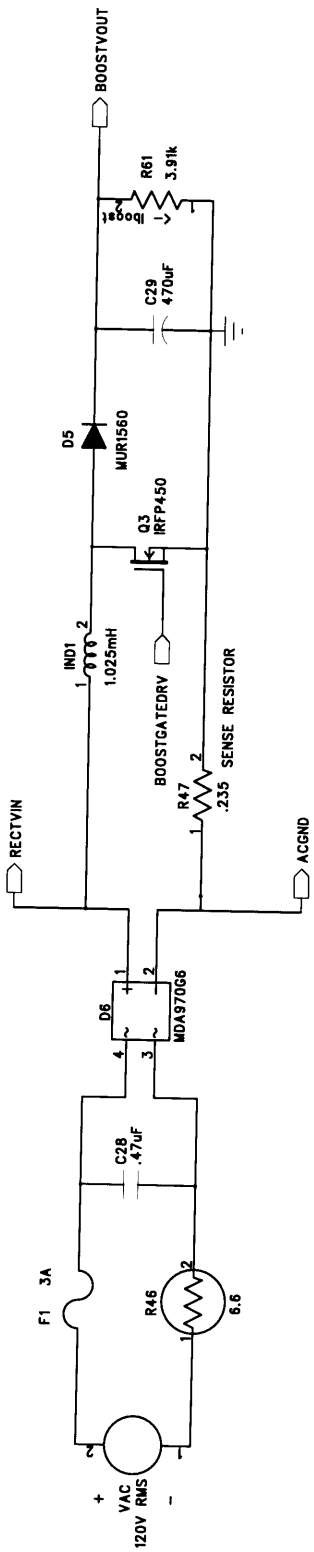

Figure D-1: Boost Converter

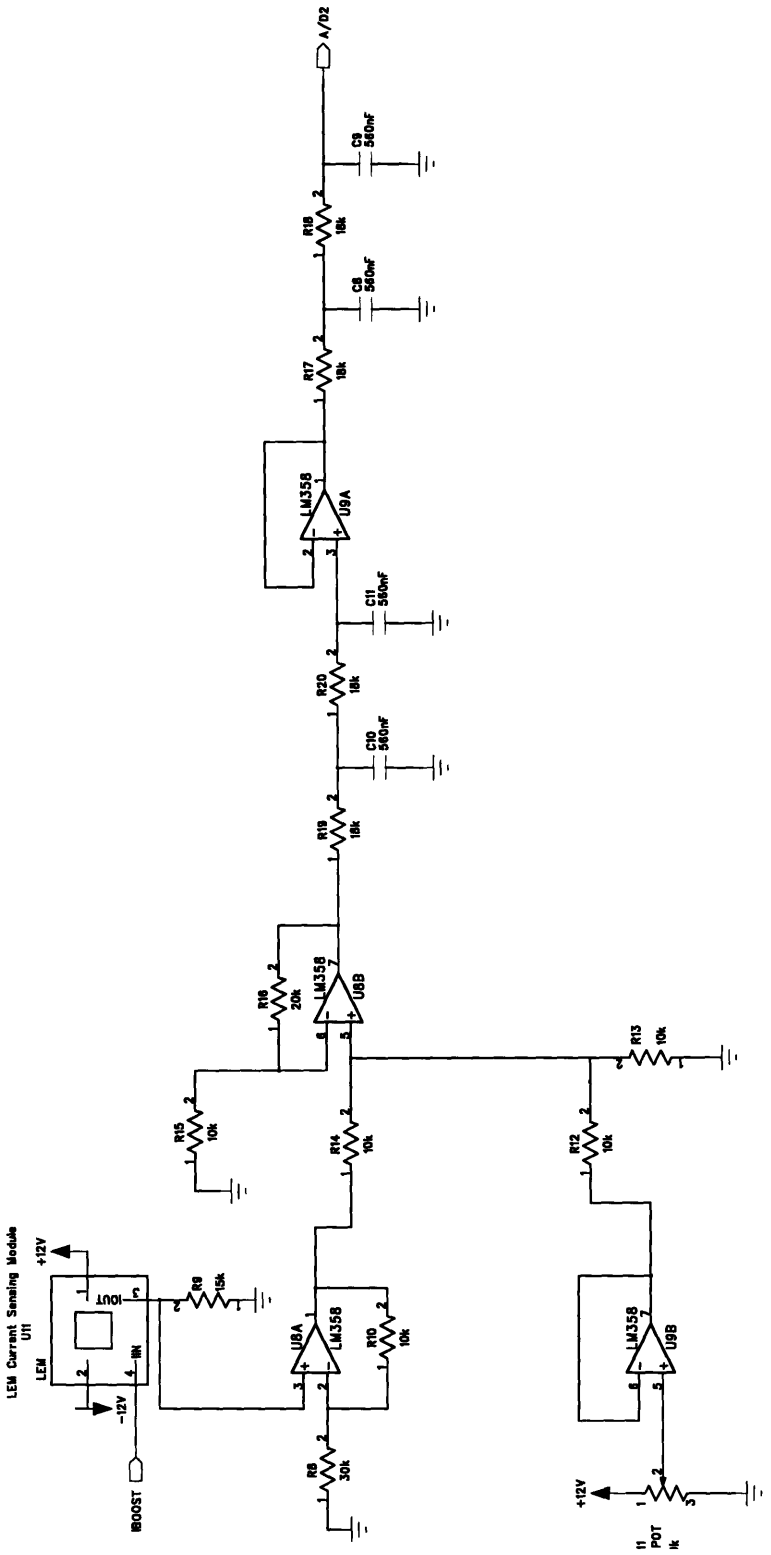

Figure D-2: Current Sensing Circuit ; Anti-aliasing Filter

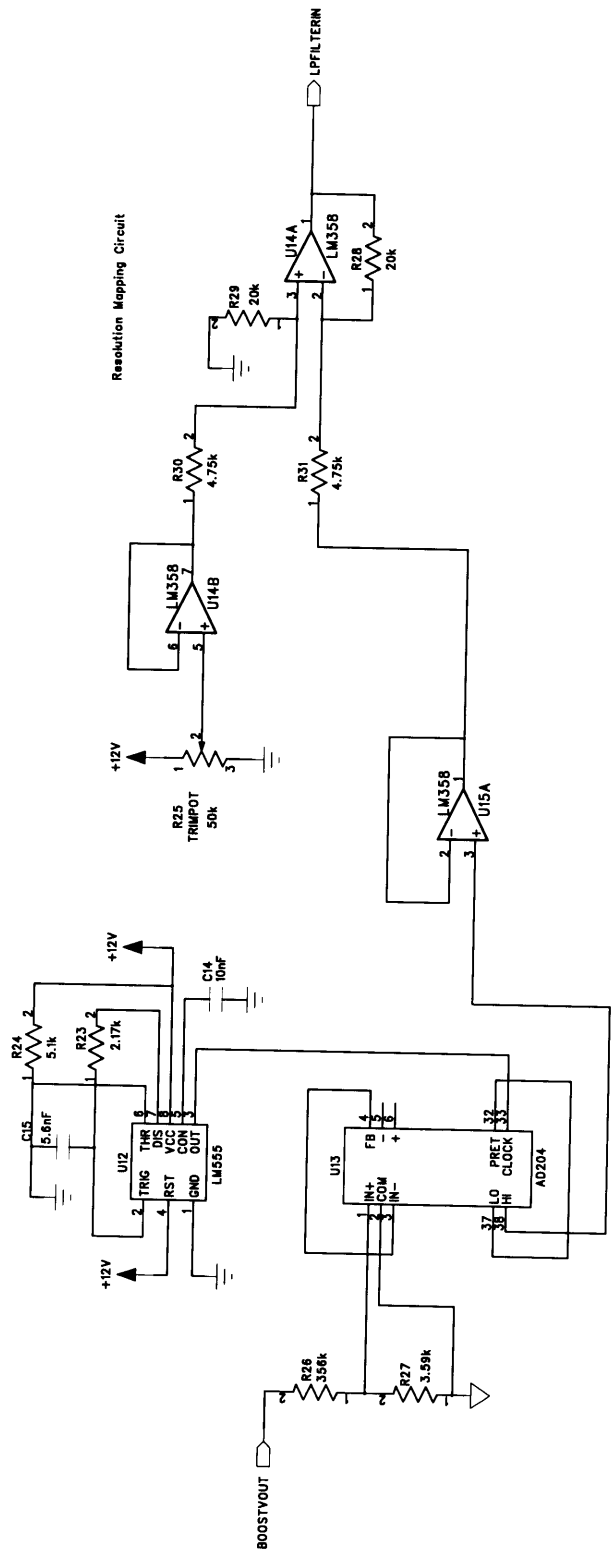

Figure D-3: Resolution Mapping of Scaled Boost Converter Output

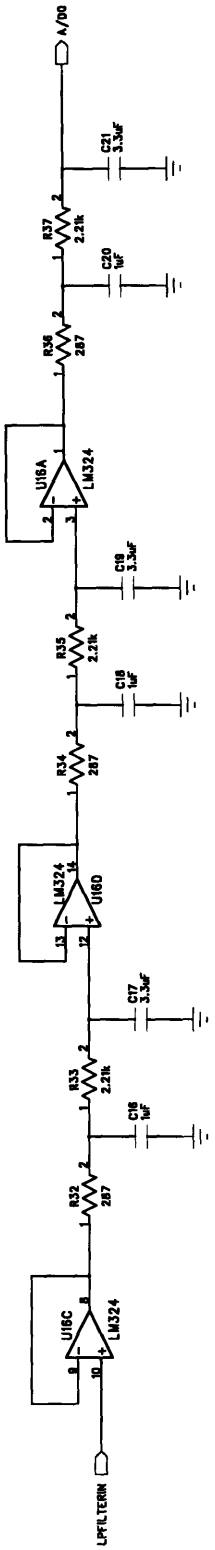

Figure D-4: Scaled Boost Converter Output Anti-aliasing Filter

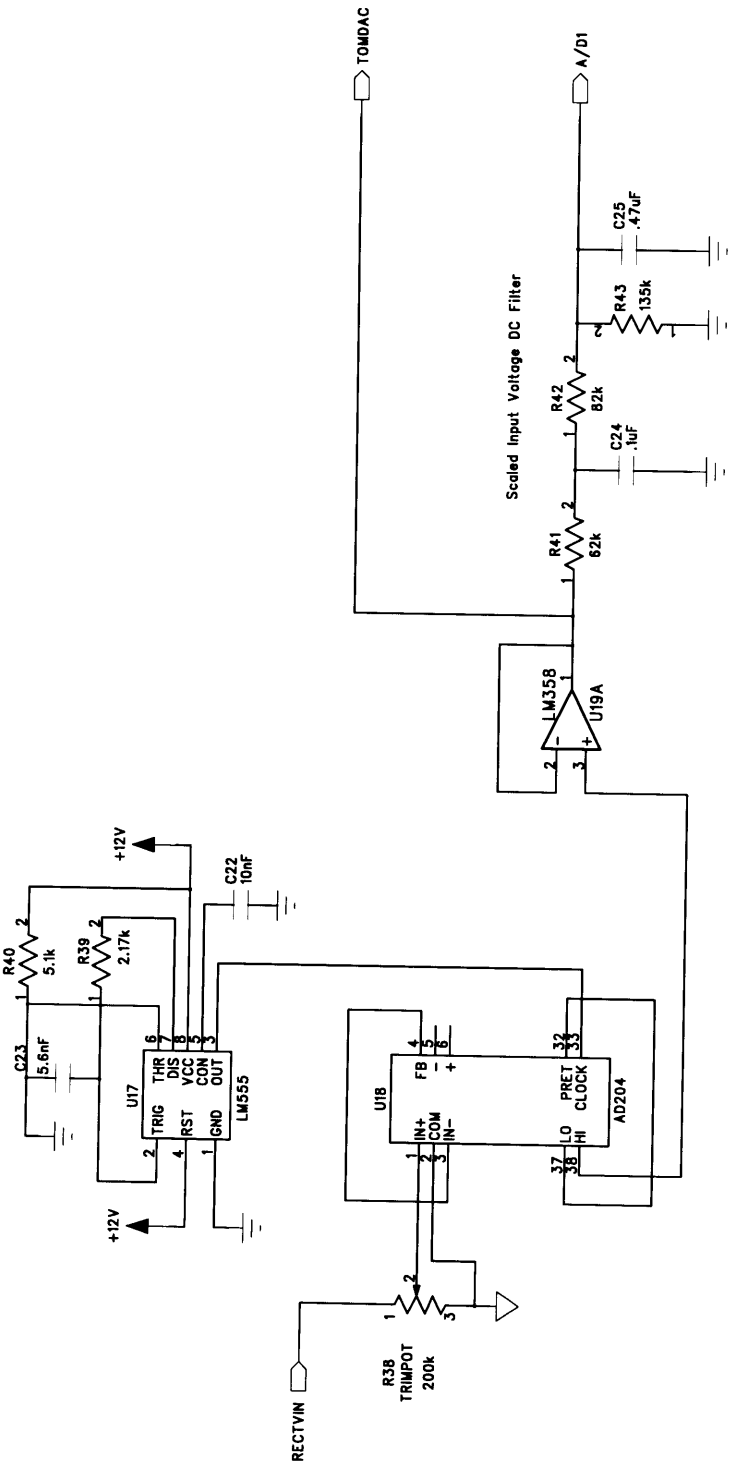

Figure D-5: Input Voltage DC Filter and Buffer

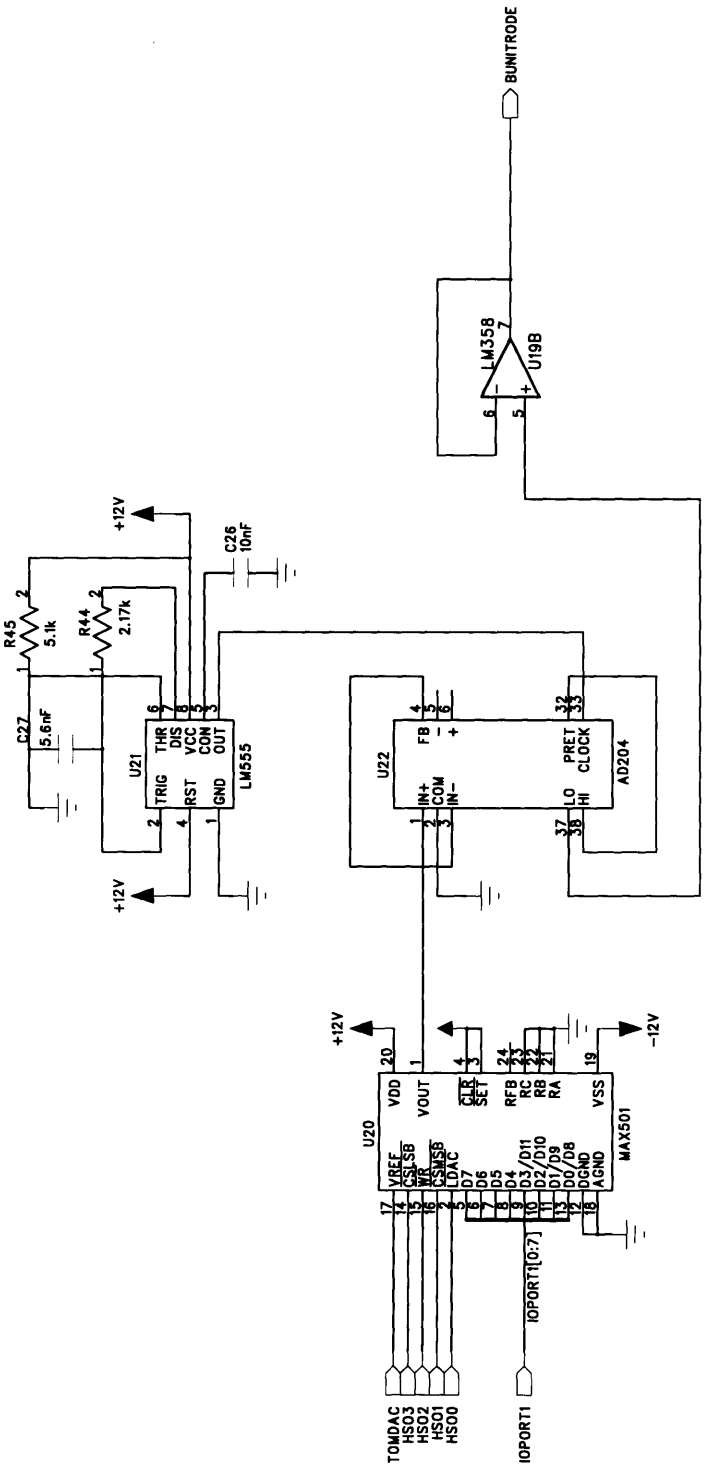

Figure D-6: MDAC Circuit

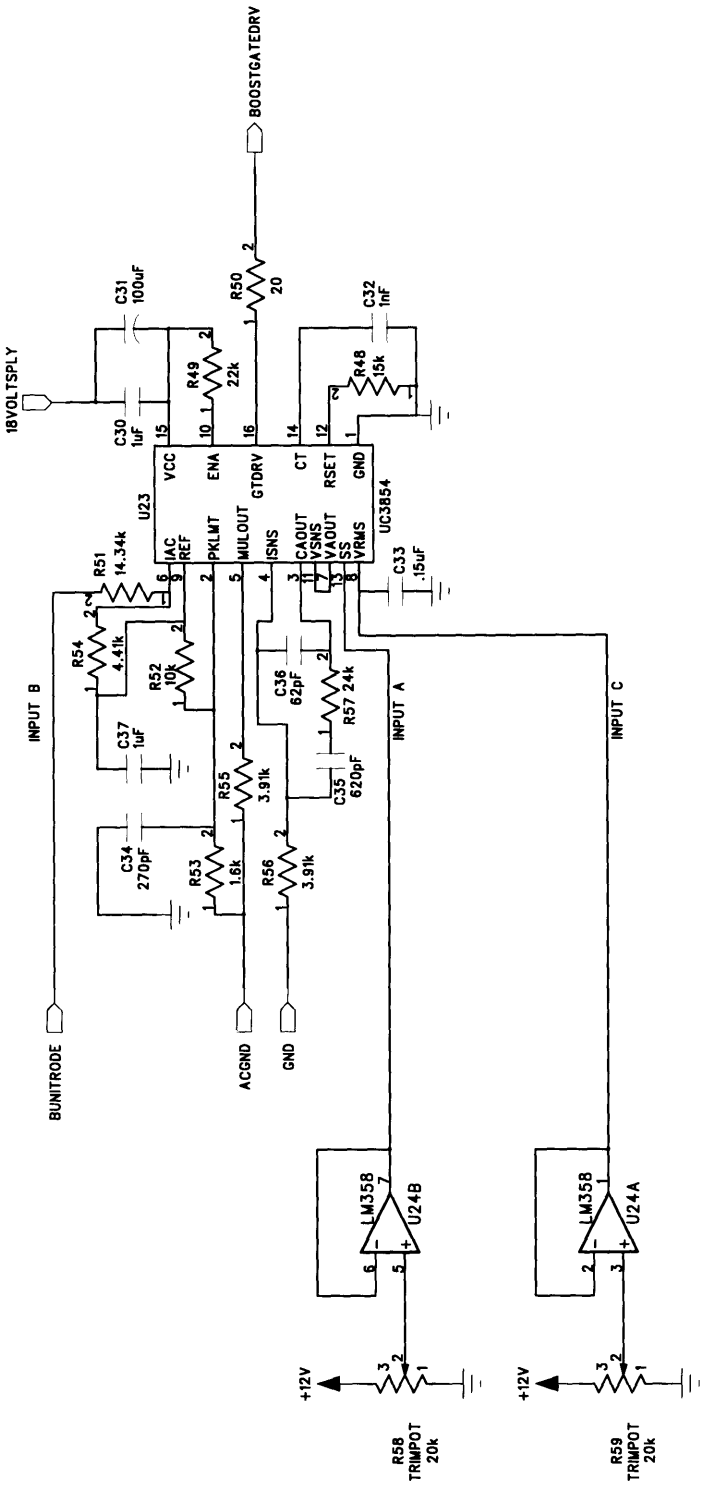

Figure D-7: Power Factor Correction Circuit

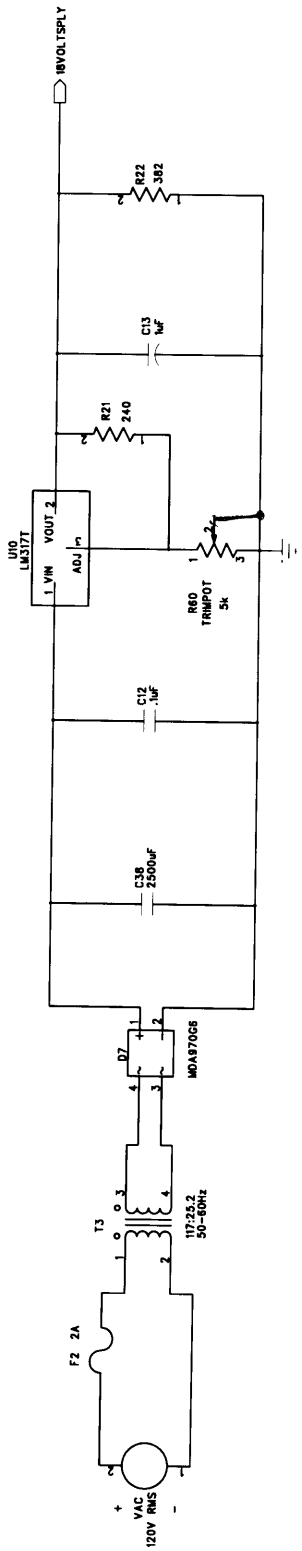

Figure D-8: 18 Volt Power Supply for UC3854

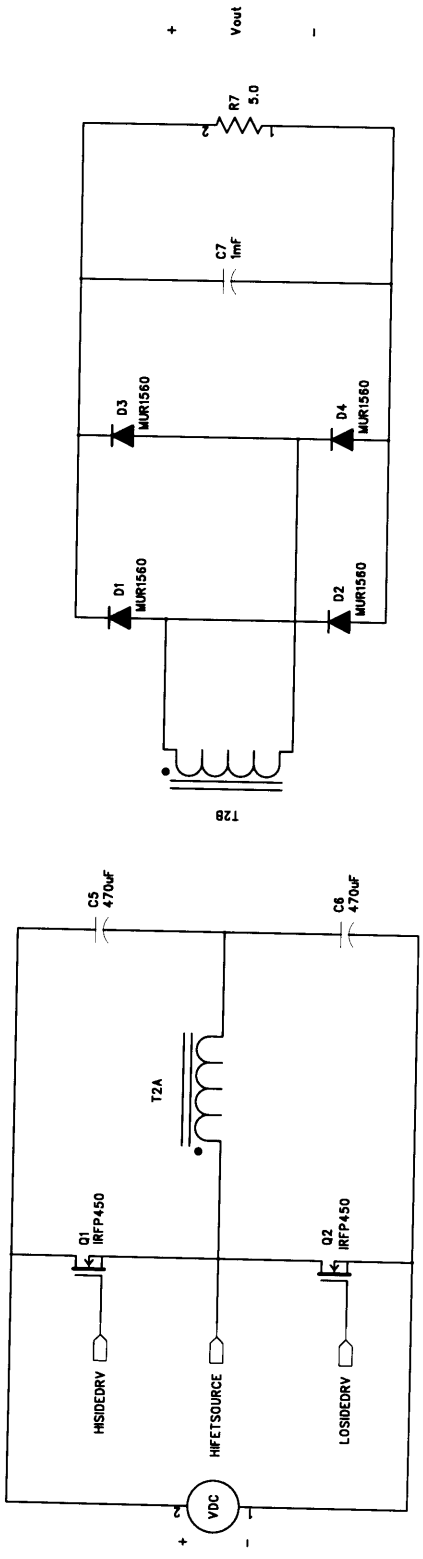

Figure D-9: DC/AC Inverter ; Rectifier

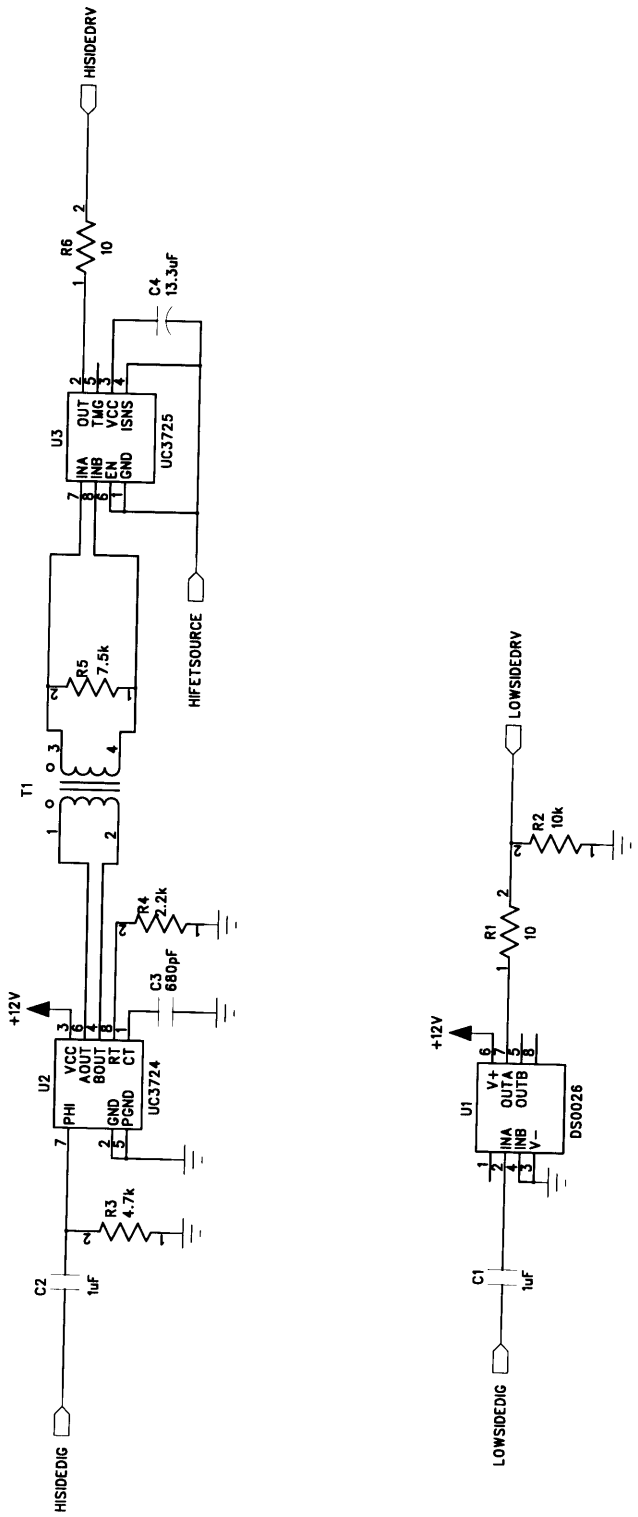

Figure D-10: Low and High Side FET Drivers

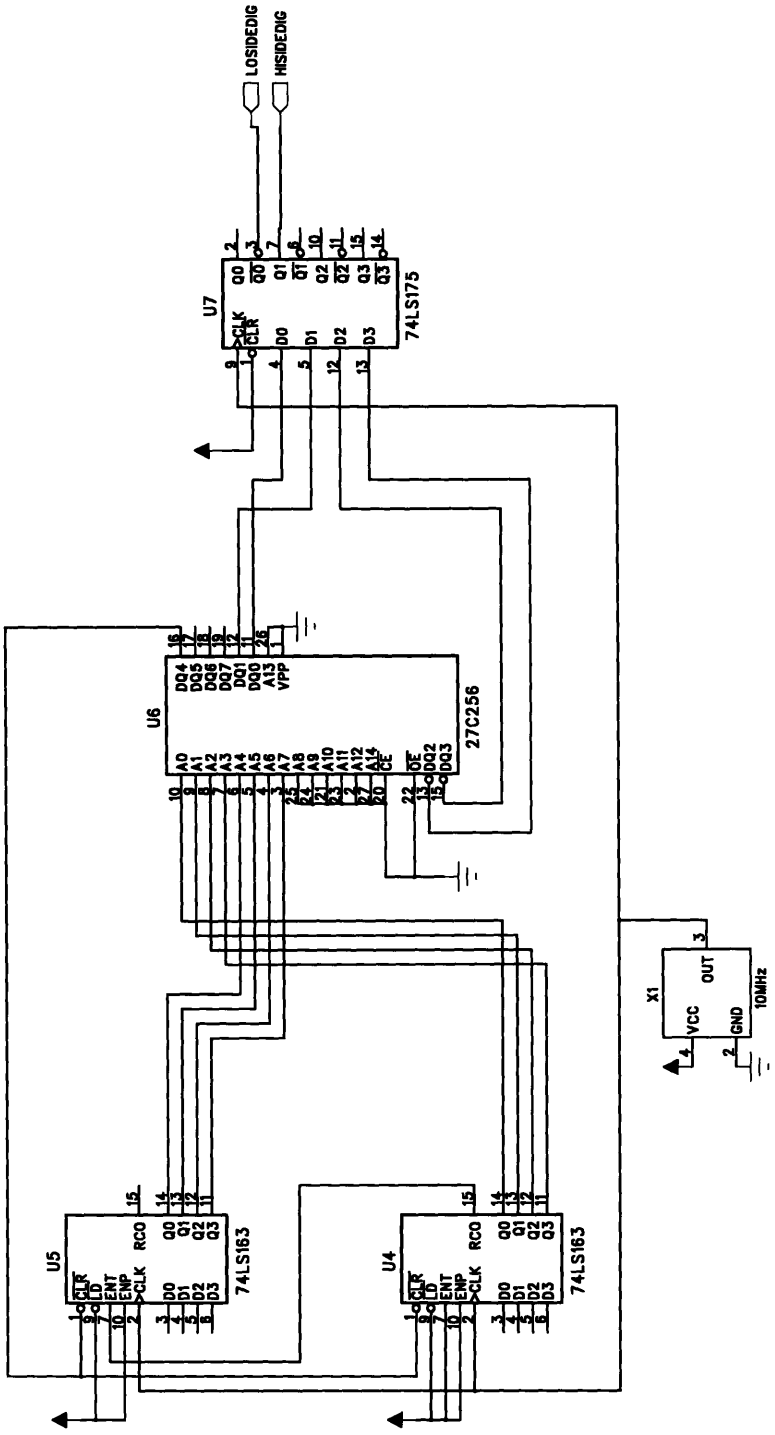

Figure D-11: Digital Switching Pattern Generator

Appendix E

Software Listing

This chapter contains MATLAB and C code for simulation and execution.

E.1 Execution Code in C

The first program listed, BASSMAN2.C, is the complete closed loop UPF current control. The procedure `software_timer` in the listing contains the implementation for changing the current reference over time for the single step upwards and downwards experiment. Following the listing will be replacement code from the oscillatory and ramping current reference experiments. The final program, BASSMAN1.C, is the UPF voltage loop controller.

```
/*****************************************************************/
/*                  Micro-Controller C Code                      */
/* This code implements the PI controller for the UPF.           */
/* Features include anti-windup, command saturation, soft startup,  */
/* and fixed command in steady-state.  Two interrupts - timer and   */
/* A/D conversion done coordinate the controlling action.        */
/* The current controller is PI, as is the voltage controller.   */
/* A delay models the voltage loop response in the domain of the */
/* current loop, which enables using simple linear state         */
/* variables for the output current and voltage.                 */
/*                                                               */
```

```
/* This file implements a simple current step response experiment.      */
/*************************************************************************/

/* Reserve locations used by the EV80C196KB. */

register char apple[9];
#pragma locate (apple = 0x30)

/* Specify model to be KB. */

#pragma model(kb)

/* Specify Interrupt Service Routines for interrupts excepted. */

#pragma interrupt (software_timer = 5)
#pragma interrupt (analog_conversion_done = 1)

#include<80C196.h>

/*** Defines ***/

/* Define the gains for the system. */
/* These digital gains are scaled versions of the actual discrete      */
/* gains.  Thus the equations are converted to containing digital      */
/* state variables which the computer can control.                     */

/* Voltage loop gains. */
/* We will multiply by h21 and right shift by h22. */

#define h1 152
#define h21 973
#define h22 8

/* Current loop gains. */
/* We will multiply by curh2 and right shift by 10. */
```

```
#define curh1 4
#define curh2 4497

/* Power coefficient for feed forward action.  */
/* We will multiply by PowerCoef1 and right shift by PowerCoef2. */

#define PowerCoef1 220
#define PowerCoef2 0

/* When count2 is STEADYSTATE, then the output voltage level */
/* has remained within the desired band enough consecutive  */
/* times for us to take the average of the last STEADYSTATE commands.*/

#define STEADYSTATE 85

/* Offset for getting rid of hack in read-in value vo. */
/* This value is artificially added to the read-in vo value for */
/* proper interpretation of that number. */

#define OFFSET 2120

/* Voltage readings of < VOTOOLOW result in max k command 4095. */

#define VOTOOLOW 2207

/* Voltage readings of > VOTOOHI result in shutting off controller. */

#define VOTOOHI 9610000

/* VRTOOHI squared is VRTOOHI, the max voltage reference. */
/* We will not use this value in this code, but if we were,
   the value 9610000 is equivalent to 371 Volts. */

#define VRTOOHI 9610000

/* The ref. squared voltage for 169.7 Volts. */
```

```
#define NOMVR 2011927

/* The equivalent of 267 Volts in digital volt squared space.          */
/* Final value for Vod^2 when soft starting in uncontrollable region.  */
/* When the voltage A/D conversion results, after squaring, in         */
/* values less than ENDOFSTART, then we are in open loop since         */
/* we don't really know what the output voltage is, and increment      */
/* the command k for the inner current loop.                           */
/* When the values are greater than ENDOFSTART, then we enter the      */
/* closed loop portion of softstart, and increment ir, the current ref. */

#define ENDSOFTSTART 4977361

/* Final current reference value for soft starting.                    */
/* Refers to 76.7 mA, which corresponds to 300 V for 3.91k load res.   */

#define FINALSOFTSTART 330

/* The HACK defines are values loaded into the voltage and current     */
/* references and accumulators when, during soft start, control        */
/* changes from open loop to closed loop.                              */
/* This happens when the output voltage reaches the resolution mapping */
/* region.  The values here have been shown experimentally to provide  */
/* a smooth transition from open loop to closed loop during soft start. */

/* For the 290 Volt hack cutoff, HACKIR is the digital i reference     */
/* assuming 3.91kOhm load.                                             */

#define HACKIR 319

/* Voltage ref. for the hack voltage (290 Volts).  */

#define HACKVR 5875496

/* When in soft starting the hack voltage is reached, the current      */
```

```
/* accumulator must be "forced" to be what it would be as if we were */
/* in steady state at the hack voltage. */

#define HACKCURACC 552
#define HACKACC 3693936

/* The output current is sampled every half-line cycle, since the */
/* voltage loop requires it to calculate the power for the feed */
/* forward term. However, the current loop operates much slower */
/* than the voltage loop. In fact, the current loop performs */
/* a PI calculation only once for every VOLTITER voltage loop */
/* PI calculations. Therefore only when count3 equals VOLTITER */
/* will the output current A/D conversion complete ISR perform */
/* a PI calculation. */

#define VOLTITER 50

/* When count4 is CURSSCNT, then the output current level */
/* has remained within the desired band enough consecutive */
/* times for us to take the average of the last CURSSCNT commands. */
/* rather than the result of a PI calculation. */

#define CURSSCNT 10

/* Current readings of > ITOOHI result in shutting off controller. */

#define ITOOHI 843

/* Gdig term. */
/* Part of the scaling of the digital gains, this number appears */
/* in the division by the input voltage rms. In this case, */
/* Gdig = .5. But we will divide by 2 instead since we have */
/* only integer arithmetic available. */

#define Gdig 2
```

```
/* Max accumulator value (digital). */
/* The digital accumulator, acc, will not be less than this */
/* number.  This number represents the accumulator value for max command */
/* when there is no steady-state error.  */

#define ACCMAX 36762754
#define CURACCMAX 761

/*** Variables.  ***/

/* v1 and v2 combine to form the 10 bit AD values assigned to vo, vi, */
/* and iin.  */

register unsigned int v1;
register unsigned int v2;
register unsigned int vo;
register unsigned int vi;
register unsigned int iin;

/* Variable steady-state zones.                                    */
/* sslo and sshi define the band within which steady state can be  */
/* achieved.  outsslo and outsshi define the band outside of which */
/* a departure from steady state must be declared.                 */

/* The steady-state numbers for voltage control. */

register long sslo;
register long sshi;
register long outsslo;
register long outsshi;

/* The steady-state numbers for current control. */

register unsigned int cursslo;
register unsigned int cur… (cursshi;)
register unsigned int curoutsslo;
```

```
register unsigned int curoutsshi;

/* result - holds PI controller command */
/* check - tells which AD channel was read */
/* vr - reference voltage squared */
/* ir - reference current squared */
/* ve - voltage error */
/* ie - current error */                                          200
/* res_temp and res_t - used for computing result */
/* k_temp - used for computing power feed forward term */
/* acc_temp - used to cmopute new accumulator value */
/* acc - contains "official" accumulator value for voltage loop */
/* curacc - contains "official" accumulator value for current loop */
/* count1 - counter for initial time delay of program execution */
/* count2 - used for voltage steady-state determination */
/* count3 - used for knowing when to apply current loop control */
/* count4 - used for current steady-state determination */
/* count5 - used to determine when to switch current references */ 210
/* flag - 1 if in voltage steady-state, 0 if not */
/* curflag - 1 if in current steady-state, 0 if not */
/* softflag - 1 if done with soft start, 0 if still in soft start */
/* sum - contains sum of last number of commands for averaging */
/* vrsum - contains sum of last number of vr values for avging */
/* dum1 - used as a dummy for band control */

register unsigned int result;
register unsigned char check;
register long vr;                                                 220
register long ir;
register long ve;
register long ie;
register long res_temp;
register long res_t;
register long k_temp;
register long acc_temp;
register long acc;
```

```
register long curacc;
register unsigned int count1;                                    230
register unsigned char count2;
register unsigned int count3;
register unsigned int count4;
register unsigned int count5;
register unsigned char flag;
register unsigned char curflag;
register unsigned char softflag;
register long sum;
register long vrsum;
register long dum1;                                              240


/* hso_command register */
/* Commands for accessing the timers and High-Speed Outputs. */
/* Bits for hso_command : */
/* 7  : CAM_LOCK    */
/* 6  : TIMER_SEL   */
/* 5  : PIN_CMD     */
/* 4  : HSPOINT_ENA */
/* 3-0 : CMD_TAG    */                                           250
/* Bit 7   : always zero - clear command from CAM after execution. */
/* Bit 6   : always zero - always use timer 1. */
/* Bit 5   : 1 - Set relevant pin(s). */
/*         : 0 - Clear relevant pin(s). */
/* Bit 4   : 1 - Generate an interrupt. */
/*         : 0 - Do not generate an interrupt. */
/* Bits 3-0 : 0 - Switch High-Speed Output 0 */
/*          : 1 - Switch High-Speed Output 1 */
/*          : 2 - Switch High-Speed Output 2 */
/*          : 3 - Switch High-Speed Output 3 */                  260
/*          : 4 - Switch High-Speed Output 4 */
/*          : 5 - Switch High-Speed Output 5 */
/*          : 6 - Switch High-Speed Outputs 0 and 1 */
/*          : 7 - Switch High-Speed Outputs 2 and 3 */
```

```
/*        : 8 - Program Software Timer 0 */
/*        : 9 - Program Software Timer 1 */
/*        : A - Program Software Timer 2 */
/*        : B - Program Software Timer 3 */
/*        : C - Switch High-Speed Outputs 0, 1, 2, 3, 4, 5 */
/*        : D - Reserved; do not use */                                    270
/*        : E - Reset Timer 2 */
/*        : F - Start an A/D Conversion */

/* hso_time register */
/* Specifies time at which an HSO command is to be executed. */
/* Bits for hso_time : */
/* 15-8 : HSO_TIME(HI) */
/* 7-0  : HSO_TIME(LO) */

/* ad_command register */                                                  280
/* Selects the A/D channel number to be converted. */
/* Bits for ad_command : */
/* 7   : Reserved    */
/* 6   : Reserved    */
/* 5   : Reserved    */
/* 4   : AD_MODE     */
/* 3   : GO          */
/* 2-0 : AD_CHAN_SEL */
/* Bits 7-5 : Reserved */
/* Bit 4    : 1 - 8-bit conversion */                                      290
/*          : 0 - 10-bit conversion */
/* Bit 3    : 1 - start immediately */
/*          : 0 - HSO initiates conversion */
/* Bits 2-0 : Channel select */

/* ad_result_hi register */
/* High 8 bits of result of A/D conversion. */
/* Bits for ad_result_hi : */
/* 7-0 : AD_HI - high 8 bits of A/D conversion result */
                                                                           300
```

```
/* ad_result_lo register */
/* Low 2 bits of result of A/D conversion. */
/* Also A/D conversion status. */
/* Bits for ad_result_lo : */
/* 7-6 : AD_1,0 - low 2 bits of A/D conversion result */
/* 5   : Reserved. */
/* 4   : AD_MODE_ST */
/* 3   : AD_STATUS */
/* 2-0 : AD_CHAN_NUM - tells which of 8 A/D channels */
/*       just completed.                              */
/* Bits 7-6 : Low two bits of A/D conversion result */
/* Bit 5    : Reserved - always write as zero       */
/* Bit 4    : 1 - 8-bit conversion                  */
/*            0 - 10-bit conversion                 */
/* Bit 3    : 1 - A/D conversion is in progress     */
/*            0 - A/D conversion is idle            */
/* Bits 2-0 : A/D channel number that was used for  */
/*            the conversion.                       */


/* Multplying DAC initialization. */
/* Set the output pins, which are connected to active low */
/* control pins on the multiplying DAC. */

void MDACinit()
{
        hso_command = 0x26;     /* set HSO 0 and 1 */
        hso_time = 0;
        hso_command = 0x27;     /* set HSO 2 and 3 */
        hso_time = 0;
}

/* Multiplying DAC write-out routine. */
/* Basically a sequence of toggling control bits accomplishes */
/* the writing of a new digital word to the DAC. */
/* Lower the CSMSB or CSLSB, lower the WR, raise the WR, */
```

```
/* raise the CSMSB or CSLSB for both the MSB and LSB. */
/* Finally, lower LDAC, then raise LDAC.  */
/* In our scheme, HSO0 - LDAC, HSO1 - CSMSB, HSO2 - WR, HSO3 - CSLSB. */
                                                                        340
void MDACwrite()
{
        int j, temp;

        /* MSB */

        ioport1 = result>>8;                    /* MSB */

        temp = ios0;
        wsr = 15;       ios0 = temp & 0xfd;     wsr = 0;                350

        temp = ios0;
        wsr = 15;       ios0 = temp & 0xf9;     wsr = 0;

        temp = ios0;
        wsr = 15;       ios0 = temp | 0x04;     wsr = 0;

        temp = ios0;
        wsr = 15;       ios0 = temp | 0x02;     wsr = 0;
                                                                        360
        /* LSB */

        ioport1 = result & 0xf0;                /* LSB */

        temp = ios0;
        wsr = 15;       ios0 = temp & 0xf7;     wsr = 0;

        temp = ios0;
        wsr = 15;       ios0 = temp & 0x03;     wsr = 0;
                                                                        370
        temp = ios0;
        wsr = 15;       ios0 = temp | 0x04;     wsr = 0;
```

```
        temp = ios0;
        wsr = 15;          ios0 = temp | 0x08;          wsr = 0;

        /* LDAC */

        temp = ios0;
        wsr = 15;          ios0 = temp & 0xfe;          wsr = 0;
        for (j=0; j<3; j++) ;

        temp = ios0;
        wsr = 15;          ios0 = temp | 0x01;          wsr = 0;
}


/* Timer ISR. */
/* In this routine, we start another timer, update the */
/* current reference if we need to, and initiate an    */
/* output current A/D conversion.   */
/* Occurs 120 times per second. */

void software_timer(void)
{
        /* Give some initial time for user to turn things on.      */
        /* Use count1 and loop 500 times with time equal           */
        /* to about 3 seconds./                                    */

        if (count1 < 500)
                {
                result = 0;
                MDACwrite();
                count1 = count1 + 1;
                hso_command = 0x18;
                hso_time = timer1 + 10000;
                }
        else
```

```
{
                                                                    410
/* Issue schedule next timer ISR */

hso_command = 0x18;             /* Command for another timer ISR. */
hso_time = timer1 + 8406;       /* Correct for 120 Hz. */

/* The step functionality.  */
/* Step up at 1200; step down at 1900. */

count5 = count5 + 1;
                                                                    420
if (count5 == 1200)
{
        ir = 387;
        cursslo = 385;
        cur… = 389;
        curoutsslo = 382;
        curoutsshi = 392;
}
if (count5 == 1900)
{                                                                   430
        ir = 330;
        cursslo = 328;
        curs… = 332;
        curoutsslo = 325;
        curoutsshi = 335;
}

/* Begin output current A/D Conversion. */

ad_command = 10;            /* Channel 2 */                         440
}
}
```

```
/* A/D processing routine */
/* All three A/D conversion complete ISRs are here. */

void analog_conversion_done(void)
{
                                                                    450
check = ad_result_lo;

/* See if it is channel 0. */

if ((check & 7) == 0)
{
v1 = (unsigned int) ad_result_hi;
v1 = v1 << 2;
v2 = (unsigned int) ad_result_lo;
v2 = v2 >> 6;                                                       460
vo = v1 + v2 + OFFSET;
                        /* To measure a factor of vo squared,          */
                        /* we add OFFSET to the 10 bit digital value.  */

dum1 = (long) vo * (long) vo;               /* Vo squared      */

/* First see if we are still in open loop soft start, and      */
/* act accordingly.                                            */
/* If we are still in soft start, then we will preload         */
/* the relevant parameters with values to assume when we move out */    470
/* of soft start.  If we not still in soft start, then set     */
/* the soft start flag. */

if (dum1 < ENDSOFTSTART)
{
        result = result + 3; /* Increment command */
        MDACwrite();        /* Write command     */
        softflag = 0;       /* Reset flag        */
        ir = HACKIR;
        curacc = HACKCURACC;                                        480
```

```
        vr = HACKVR;
        acc = HACKACC;
        return;          /* That's it for this ISR.        */
                         /* Soft start means open loop, so */
                         /* no PI control.                 */
}
else softflag = 1;       /* Otherwise set flag. */

/* Check to make sure vo is not too high. */
                                                                    490
if (dum1 > VOTOOHI)
{
        result = 0;
        MDACwrite();             /* Write out a zero.       */
        while(1);                /* Kill the program.       */
}

/* Steady-state action.  */

if ((dum1 < sshi) && (dum1 > sslo) && (count2 < STEADYSTATE))       500
{
        /* We're in the s-s band, but we haven't reached the right */
        /* number of repetitions for s-s.  */

         count2 = count2 + 1;
         sum = sum + result;
 }

/* If in steady-state for STEADYSTATE cycles, set flag to 1 to indicate */
/* officially in steady-state and can fix output command based  */      510
/* on previous STEADYSTATE commands.                            */

 if ((count2 == STEADYSTATE) && (dum1 < sshi) && (dum1 > sslo))
        {
                flag = 1;
        }
```

```
else
if ((dum1 > outsshi) || (dum1 < outsslo))
        {                                                                    520
        /* We're not only not in steady-state, but we've stepped out */
        /* of the safety hysteresis range.  Start over. */

        flag = 0;
        count2 = 0;
        sum = 0;
        }

/* Now act on the flag - send out an average command if flag is set. */
/* Otherwise start a channel 1 A/D conversion. */                        530

if (flag == 1)
        {
        /* We're in steady-state, so send out the average command. */
        /* Do NOT start a channel 1 vin A/D command. */

        result = (unsigned int) (sum/STEADYSTATE);
        MDACwrite();

        }                                                                    540

else
        {
        /* Only start vin A/D conversion if not in steady-state band. */

        ad_command = 9;   /* Start channel 1.  */
        }

}       /* End of channel 0 A/D check.  */
                                                                             550
/* See if it is channel 1. */
```

```
else if ((check & 7) == 1)
        {
        v1 = (unsigned int) ad_result_hi;
        v1 = v1 << 2;
        v2 = (unsigned int) ad_result_lo;
        v2 = v2 >> 6;
        vi = v1 + v2;
                                                                        560
        /* Too low output - jam on the power, dude. */
/*      if (vo < VOTOOLOW) result = 4095; */

                /*** The actual PI control calculation. ***/
                ve = dum1;                              /* Vo squared    */
                ve = vr - ve;                           /* Error         */
                res_temp = ve * h1;                     /* Gain1*error   */
                res_t = acc >> h22;
           res_t = res_t * h21;             /* Gain2*acc    */
                res_temp = res_temp + res_t;            /* Sum           */      570
                res_t = (long) vi * (long) vi / (long)Gdig;
                                                        /* Scale Vi^2    */

                res_temp = res_temp / res_t;            /* Final calc.: */
                                                        /* Div by vi^2. */

           /* Now add in the power feed foward term. */
           k_temp = PowerCoef1 * vo * iin;      /* PowerGain *  */
           k_temp = k_temp >> PowerCoef2;       /* input power. */
           k_temp = k_temp / (res_t * (long)Gdig); /* Div. by vi^2 */       580
           res_temp = res_temp + k_temp;

                /* In case of too high command. */
                if (res_temp > 4095)
                        result = 4095;
                  else
                  {
                        /* So we're operating normally at this point.  *
```

```
                * We could check for flag==0 also here,       *
                *  but channel 1 doesn't get set in motion    *
                * if flag==1.  It should be flag==0 already.  */

                acc_temp = acc + ve;            /* Change acc.    */

                if (acc_temp > ACCMAX)          /* Max accum?     */
                        acc = ACCMAX;   /* Note: ACCMAX > 0.      */

                else if (acc_temp < 0)          /* acc negative?*/
                        acc = 0;

                else acc = acc_temp;            /* We're okay.    */

                if (res_temp < 0)
                        result = 0;
                else
                {
                /* Set result to the calculation answer. */
                result = (unsigned int) res_temp;
                }
        }

/* Here is the big write out command. */
/* If flag is 1, we're in steady-state, so send out
   the average.  Else send out the calculated result. */

if (flag == 1)
        /**********************************
         We will never get here if channel 0's
         code doesn't trigger a channel 1 conversion when
         flag is 1. This code is only beautific.***********/
        {
        result = (unsigned int) (sum/STEADYSTATE);
        MDACwrite();
        }
```

```
        else
        {
                /* So we have a result, and we're not in steady-state      */
                /* band.  Write out the result to the DAC.                 */
                MDACwrite();
        }                                                                    630


}/* End of check for channel 1.  */

/* See if it is channel 2. */

else if ((check & 7) == 2)
{
        v1 = (unsigned int) ad_result_hi;
        v1 = v1 << 2;                                                        640
        v2 = (unsigned int) ad_result_lo;
        v2 = v2 >> 6;
        iin = v1 + v2;

        /* Check for too high current. */

        if (iin > ITOOHI)
        {
                result = 0;
                MDACwrite();          /* Write out a zero.       */          650
                while(1);             /* Kill the program.       */
        }

        /* See if we should apply current control here. */
        /* First check out current steady-state band action. */

        if ((iin < cursshi) && (iin > cursslo) && (count4 < CURSSCNT))
        {
                /* We're in the s-s band, but we haven't reached the right */
                /* number of repetitions for s-s.  */                        660
```

```
        count4 = count4 + 1;
        vrsum = vrsum + vr;
    }

    if ((count4 == CURSSCNT) && (iin < cursshi) && (iin > cursslo))
        {
            /* Steady-state has been achieved. */

            curflag = 1;                                                    670
        }
    else
    if ((iin > curoutsshi) || (iin < curoutsslo))
        {
        /* Out of the band.  Start over. */

        curflag = 0;
        count4 = 0;
        vrsum = 0;
        }                                                                   680

    count3 = count3 + 1;

    /* Only do a current loop calculation if we have done          */
    /* VOLTITER voltage loop calculations                          */
/* since the current loop runs slower than the          */
/* voltage loop.                                              */

    if (count3 == VOLTITER)
    {                                                                       690
        /* First if we are in soft start, then incr. ir.  */

        if ((softflag == 1) && (ir + 1) < FINALSOFTSTART)
            ir = ir + 11;

        count3 = 0;
```

```
/* If we are in steady state, then it's easy! */

if (curflag == 1)                                                    700
        vr = vrsum / CURSSCNT;

else {          /* Not in steady state.  */

/*** The actual PI control calculation. ***/
ie = (long) iin;
ie = ir - ie;                           /* Error          */
res_temp = ie * curh1;                  /* Gain1 * error*/
res_t = (curacc * curh2) >> 10;         /* Gain2 * acc    */
res_temp = res_temp + res_t;            /* Sum            */        710

/* In case of too high voltage command. */
/*      if (res_temp > VRTOOHI)
        vr = VRTOOHI; */

        acc_temp = curacc + ie;         /* Change acc.    */

        if (acc_temp > CURACCMAX)       /* Max accum?     */
                curacc = CURACCMAX;     /* ACCMAX > 0  */
                                                                     720
        else if (acc_temp < 0)          /* acc negative?*/
                acc = 0;

        else curacc = acc_temp;         /* We're okay.    */

        if (res_temp < 0)
                vr = NOMVR;

        else
        /* Set vr to the calculation answer. */                      730

                vr = res_temp * res_temp;
```

```
                }

        /* So at this point we have set vr.                                    */
        /* It is time to set the voltage s-s bands.                            */
        /* We could use the linear version of the calculated vr                */
        /* (res_temp) by adding and subtracting from and then                  */
        /* squaring to find the s-s band values.  Instead,                     */
        /* we will add and subtract a percentage of the final                  */
        /* vr from vr itself.                                                  */

        res_temp = vr >> 7;             /* Divide vr by 128.                   */
        sslo = vr - (res_temp*1);       /* Add and subtract accordingly.*/
        sshi = vr + (res_temp*1);
        outsslo = vr - (2*res_temp*1);
        outsshi = vr + (2*res_temp*1);

        }/* End of if count3 == VOLTITER.   */

      /* Always start the output voltage control.   */

      ad_command = 8;         /* Channel 0. */

 }/* End of check for channel 2.  */

} /* End of analog_conversion_done ISR.  */


/* The main routine.  */

main()
{
        /* Initialize lots of stuff.  */

        count1 = 0;
        count2 = 0;
        count3 = 0;
```

```
    count4 = 0;
    count5 = 0;                                                          770
    flag = 0;
    curflag = 0;
    softflag = 0;
    sum = 0;
    result = 0;
    vr = NOMVR;    /* Corresponds to 169.7 V on output.          */

    /* The initial values of the current steady state variables          */
    /* will be useful when the soft start finally reaches ir = 330. */
                                                                         780
    ir = 0;
    cursslo = 328;
    cur… = 332;
    curoutsslo = 325;
    curoutsshi = 335;

    acc = 0;
    curacc = 0;

    MDACinit();                                                          790

    result = 0;                  /* Initial zero command.          */
    MDACwrite();          /* Write command.                 */

    ioport1 = 0;          /* Write zero to ioport           */

    int_mask = 0x22;      /* Initialize interrupts          */
    int_pending = 0;
    hso_command = 0x18;    /* Schedule the first timer ISR */
    hso_time = timer1 + 6254;                                            800
/*  ad_command = 8; */
    enable();                    /* Enable interrupts.             */

    while(1);                    /* Let interrupts do their work.*/
```

```
} /* End main */
```

```
/* The oscillation functionality.  */
/* Step up at 1200; step down at 1900. */

if (count5 == 1200)
{
        ir = 387;
        cursslo = 385;
        curshi = 389;
        curoutsslo = 382;
        curoutsshi = 392;
}
if (count5 == 1900)
{
        ir = 330;
        cursslo = 328;
        curshi = 332;
        curoutsslo = 325;
        curoutsshi = 335;
}
if (count5 == 2600)
{
        ir = 387;
        cursslo = 385;
        curshi = 389;
        curoutsslo = 382;
        curoutsshi = 392;
}
if (count5 == 3300)
{
        ir = 330;
        cursslo = 328;
```

```
        cursshi = 332;
        curoutsslo = 325;
        curoutsshi = 335;
}
if (count5 == 4000)
{
        ir = 387;
        cursslo = 385;
        cursshi = 389;
        curoutsslo = 382;
        curoutsshi = 392;
}
if (count5 == 4700)
{
        ir = 330;
        cursslo = 328;
        cursshi = 332;
        curoutsslo = 325;
        curoutsshi = 335;
}
if (count5 == 5400)
{
        ir = 387;
        cursslo = 385;
        cursshi = 389;
        curoutsslo = 382;
        curoutsshi = 392;
}
if (count5 == 6100)
{
        ir = 330;
        cursslo = 328;
        cursshi = 332;
        curoutsslo = 325;
        curoutsshi = 335;
}
```

```
    /* The ramping reference functionality.  */

    if (count5 > 2600)
    {
    ir = 387-((count5-2600)>>7);
    cursslo = ir-2;
    cursshi = ir+2;
    curoutsslo = ir-5;
    curoutsshi = ir+5;                                                    10
    }
```

```
/*************************************************************************/
/*                  Micro-Controller C Code                              */
/* This code implements the PI controller for the UPF.                   */
/* Features include anti-windup, command saturation, soft startup,       */
/* and fixed command in steady-state.  Two interrupts - timer and        */
/* A/D conversion done coordinate the controlling action.                */
/* The voltage controller is PI.                                         */
/* This program runs the voltage loop only.                    */
/*                                                             */
/* This file implements step response.                         */        10
/*************************************************************************/

/* Reserve locations used by the EV80C196KB. */

register char apple[9];
#pragma locate (apple = 0x30)

/* Specify model to be KB. */

#pragma model(kb)                                                         20

/* Specify Interrupt Service Routines for interrupts excepted. */
```

```
#pragma interrupt (software_timer = 5)
#pragma interrupt (analog_conversion_done = 1)

#include<80C196.h>

/*** Defines ***/

/* Define the gains for the system. */
/* These digital gains are scaled versions of the actual discrete */
/* gains.  Thus the equations are converted to containing digital */
/* state variables which the computer can control. */

/* Voltage loop gains. */
/* We will multiply by h21 and right shift by h22. */

#define h1 152
#define h21 973
#define h22 8

/* Power coefficient for feed forward action. */
/* We will multiply by PowerCoef1 and right shift by PowerCoef2. */

#define PowerCoef1 220
#define PowerCoef2 0

/* When count2 is STEADYSTATE, then the output voltage level */
/* has remained within the desired band enough consecutive */
/* times for us to take the average of the last STEADYSTATE commands.*/

#define STEADYSTATE 18

/* Offset for getting rid of hack in read-in value vo. */
/* This value is artificially added to the read-in vo value for */
/* proper interpretation of that number. */
```

```
#define OFFSET 2120

/* Voltage readings of < VOTOOLOW result in max command 4095. */

#define VOTOOLOW 2207

/* Voltage readings of > VOTOOHI result in shutting off controller. */

#define VOTOOHI 3100

/* The equivalent of 278 Volts in digital volt space.                    */
/* Final value for Vod^2 when soft starting in uncontrollable region.    */
/* When the voltage A/D conversion results, after squaring, in           */
/* values less than ENDOFSTART, then we are in open loop since           */
/* we don't really know what the output voltage is, and increment        */
/* the command k for the inner current loop.                             */
/* When the values are greater than ENDOFSTART, then we enter the        */
/* closed loop portion of softstart, and increment ir, the current ref. */

#define ENDSOFTSTART 2324

/* Final squared voltage eference value for soft starting.               */
/* Refers to 300 V. */

#define FINALSOFTSTART 6290064

/* The HACK defines are values loaded into the voltage and current       */
/* references and accumulators when, during soft start, control          */
/* changes from open loop to closed loop.                                */
/* This happens when the output voltage reaches the resolution mapping   */
/* region.  The values here have been shown experimentally to provide    */
/* a smooth transition from open loop to closed loop during soft start. */

/* Voltage ref. for the hack voltage (290 Volts).  */
/* Then final voltage while soft starting.  */
```

```
#define HACKVR 5875496

/* When in soft starting the hack voltage is reached, the current      */
/* accumulator must be "forced" to be what it would be as if we were   */
/* in steady state at the hack voltage.                                */

#define HACKVOLTACC 3693936

/* Reference voltage squared, a key number in the calculation. */
/* Should be set near the final desired vo squared as determined */
/* By the steady-state band. */

/* Current readings of > ITOOHI result in shutting off controller. */

#define ITOOHI 843

/* Gdig term. */
/* Part of the scaling of the digital gains, this number appears */
/* in the division by the input voltage rms.  In this case,      */
/* Gdig = .5.  But we will divide by 2 instead since we have     */
/* only integer arithmetic available.                            */

#define Gdig 2

/* Max accumulator value (digital). */
/* The digital accumulator, acc, will not be less than this */
/* number.  This number represents the accumulator value for max command */
/* when there is no steady-state error. */

#define ACCMAX 36762754

/*** Variables. ***/

/* v1 and v2 combine to form the 10 bit AD values assigned to vo and vi, */
/* and iin. */
```

```
register unsigned int v1;
register unsigned int v2;
register unsigned int vo;
register unsigned int vi;
register unsigned int iin;

/* result - holds PI controller command */
/* check - tells which AD channel was read */
/* vr - reference voltage squared */
/* ve - voltage error */                                          140
/* res_temp and res_t - used for computing result */
/* k_temp - used for computing power feed forward term */
/* acc_temp - used to cmopute new accumulator value */
/* acc - contains "official" accumulator value */
/* count1 - counter for initial time delay of program execution */
/* count2 - used for steady-state determination */
/* flag - 1 if in steady-state, 0 if not */
/* softflag - 1 if completed open loop soft start, 0 if not */
/* sum - contains sum of last number of commands for averaging */
/* dum1 - used as a dummy for band control */                     150

register unsigned int result;
register unsigned char check;
register long vr;
register long ve;
register long res_temp;
register long res_t;
register long k_temp;
register long acc_temp;
register long acc;                                                160
register unsigned int count1;
register unsigned char count2;
register unsigned int count3;
register unsigned char flag;
register unsigned char softflag;
register long sum;
```

```
register unsigned int dum1;

/* Variables for steady-state band limits. */
/* sslo and sshi are the min and max values for desired output */
/* voltage reading (with OFFSET already added). */
/* outsslo and outsshi are the vo values beyond which (less than and */
/* greater than, respectively) count2 is reset to 0 and we */
/* start the steady-state band procedure all over again. */

register unsigned int outsshi;
register unsigned int outsslo;
register unsigned int sshi;
register unsigned int sslo;

/* hso_command register */
/* Commands for accessing the timers and High-Speed Outputs. */
/* Bits for hso_command : */
/* 7  : CAM_LOCK    */
/* 6  : TIMER_SEL   */
/* 5  : PIN_CMD     */
/* 4  : HSPOINT_ENA */
/* 3-0 : CMD_TAG    */
/* Bit 7   : always zero - clear command from CAM after execution. */
/* Bit 6   : always zero - always use timer 1. */
/* Bit 5   : 1 - Set relevant pin(s). */
/*         : 0 - Clear relevant pin(s). */
/* Bit 4   : 1 - Generate an interrupt. */
/*         : 0 - Do not generate an interrupt. */
/* Bits 3-0 : 0 - Switch High-Speed Output 0 */
/*         : 1 - Switch High-Speed Output 1 */
/*         : 2 - Switch High-Speed Output 2 */
/*         : 3 - Switch High-Speed Output 3 */
/*         : 4 - Switch High-Speed Output 4 */
/*         : 5 - Switch High-Speed Output 5 */
/*         : 6 - Switch High-Speed Outputs 0 and 1 */
```

```
/*        : 7 - Switch High-Speed Outputs 2 and 3 */
/*        : 8 - Program Software Timer 0 */
/*        : 9 - Program Software Timer 1 */
/*        : A - Program Software Timer 2 */
/*        : B - Program Software Timer 3 */
/*        : C - Switch High-Speed Outputs 0, 1, 2, 3, 4, 5 */
/*        : D - Reserved; do not use */
/*        : E - Reset Timer 2 */                                    210
/*        : F - Start an A/D Conversion */

/* hso_time register */
/* Specifies time at which an HSO command is to be executed. */
/* Bits for hso_time : */
/* 15-8 : HSO_TIME(HI) */
/* 7-0  : HSO_TIME(LO) */

/* ad_command register */
/* Selects the A/D channel number to be converted. */                220
/* Bits for ad_command : */
/* 7   : Reserved    */
/* 6   : Reserved    */
/* 5   : Reserved    */
/* 4   : AD_MODE     */
/* 3   : GO          */
/* 2-0 : AD_CHAN_SEL */
/* Bits 7-5 : Reserved */
/* Bit 4    : 1 - 8-bit conversion */
/*          : 0 - 10-bit conversion */                               230
/* Bit 3    : 1 - start immediately */
/*          : 0 - HSO initiates conversion */
/* Bits 2-0 : Channel select */

/* ad_result_hi register */
/* High 8 bits of result of A/D conversion. */
/* Bits for ad_result_hi : */
/* 7-0 : AD_HI - high 8 bits of A/D conversion result */
```

```
/* ad_result_lo register */                                          240
/* Low 2 bits of result of A/D conversion. */
/* Also A/D conversion status. */
/* Bits for ad_result_lo : */
/* 7-6 : AD_1,0 - low 2 bits of A/D conversion result */
/* 5   : Reserved   */
/* 4   : AD_MODE_ST */
/* 3   : AD_STATUS  */
/* 2-0 : AD_CHAN_NUM - tells which of 8 A/D channels  */
/*       just completed.                                 */
/* Bits 7-6 : Low two bits of A/D conversion result   */             250
/* Bit 5    : Reserved - always write as zero         */
/* Bit 4    : 1 - 8-bit conversion                    */
/*            0 - 10-bit conversion                   */
/* Bit 3    : 1 - A/D conversion is in progress       */
/*            0 - A/D conversion is idle              */
/* Bits 2-0 : A/D channel number that was used for    */
/*            the conversion.                         */



/* Multplying DAC initialization.   */                               260
/* Set the output pins, which are connected to active low */
/* control pins on the multiplying DAC. */

void MDACinit()
{
        hso_command = 0x26;      /* set HSO 0 and 1 */
        hso_time = 0;
        hso_command = 0x27;      /* set HSO 2 and 3 */
        hso_time = 0;
}                                                                    270


/* Multiplying DAC write-out routine.   */
/* Basically a sequence of toggling control bits accomplishes */
```

```
/* the writing of a new digital word to the DAC. */
/* Lower the CSMSB or CSLSB, lower the WR, raise the WR, */
/* raise the CSMSB or CSLSB for both the MSB and LSB. */
/* Finally, lower LDAC, then raise LDAC.  */
/* In our scheme, HSO0 – LDAC, HSO1 – CSMSB, HSO2 – WR, HSO3 – CSLSB. */
                                                                        280
void MDACwrite()
{
        int j, temp;

        /* MSB */

        ioport1 = result>>8;                    /* MSB */

        temp = ios0;
        wsr = 15;       ios0 = temp & 0xfd;     wsr = 0;                290

        temp = ios0;
        wsr = 15;       ios0 = temp & 0xf9;     wsr = 0;

        temp = ios0;
        wsr = 15;       ios0 = temp | 0x04;     wsr = 0;

        temp = ios0;
        wsr = 15;       ios0 = temp | 0x02;     wsr = 0;
                                                                        300
        /* LSB */

        ioport1 = result & 0xf0;                /* LSB */

        temp = ios0;
        wsr = 15;       ios0 = temp & 0xf7;     wsr = 0;

        temp = ios0;
        wsr = 15;       ios0 = temp & 0x03;     wsr = 0;
                                                                        310
```

```
        temp = ios0;
        wsr = 15;       ios0 = temp | 0x04;     wsr = 0;

        temp = ios0;
        wsr = 15;       ios0 = temp | 0x08;     wsr = 0;

        /* LDAC */

        temp = ios0;
        wsr = 15;       ios0 = temp & 0xfe;     wsr = 0;                    320
        for (j=0; j<3; j++) ;

        temp = ios0;
        wsr = 15;       ios0 = temp | 0x01;     wsr = 0;
}


/* Timer ISR. */
/* In this routine, we start another timer, update the */
/* voltage reference if we need to, and initiate an    */                   330
/* output current A/D conversion.  */
/* Should occur 120 times per second. */

void software_timer(void)
{
        /* Give some initial time for user to turn things on.      */
        /* Use count1 and loop 500 times with time equal           */
        /* to about 3 seconds./                                    */


                                                                            340
        if (count1 < 500)
                {
                result = 0;
                MDACwrite();
                count1 = count1 + 1;
                hso_command = 0x18;
```

```
            hso_time = timer1 + 10000;
            }
    else
    {                                                                    350

/* Issue schedule next timer ISR. */

    hso_command = 0x18;          /* Command for another timer ISR. */
    hso_time = timer1 + 8406;    /* Correct for 120 Hz. */

    /* Here is where we implement the step response.    */

    count3 = count3 + 1;
                                                                         360
    if (count3 == 1200)
    {
            vr = 9054081;
            sslo = 3009;
            sshi = 3029;
            outsslo = 2999;
            outsshi = 3039;

    }
                                                                         370
    if (count3 == 1900)
    {
            vr = 6290064;
            sslo = 2508;
            sshi = 2528;
            outsslo = 2498;
            outsshi = 2538;
    }

    /* Begin output current A/D Conversion. */                          380

    ad_command = 10;                    /* Channel 2 */
```

```
        }
}


/* A/D processing routine */
/* All three A/D conversion complete ISRs are here. */
                                                                    390
void analog_conversion_done(void)
{

check = ad_result_lo;

/* See if it is channel 0. */

if ((check & 7) == 0)
{
v1 = (unsigned int) ad_result_hi;                                   400
v1 = v1 << 2;
v2 = (unsigned int) ad_result_lo;
v2 = v2 >> 6;
vo = v1 + v2 + OFFSET;
                        /* To measure a factor of vo squared,          */
                        /* we add OFFSET to the 10 bit digital value.  */
dum1 = vo;

/* First see if we are still in open loop soft start, and      */
/* act accordingly.                                            */  410
/* If we are still in soft start, then we will preload         */
/* the relevant parameters with values to assume when we move out */
/* of soft start.  If we not still in soft start, then set     */
/* the soft start flag. */

if (dum1 < ENDSOFTSTART)
{
      result = result + 3; /* increment command */
```

```
        MDACwrite();        /* Write command     */
        softflag = 0;       /* Reset flag        */                    420
        vr = HACKVR;
        acc = HACKVOLTACC;
        return;             /* That's it for this ISR.       */
                    /* Soft start means open loop, so */
                    /* no PI control.                */
}
else softflag = 1;          /* Otherwise set flag. */

/* Otherwise, check to see if we are not in open loop soft start */
/* but are still in soft start (output voltage less than target. */   430

if ((softflag == 1) && (vr < FINALSOFTSTART)) /* softflag is 1 if vo is > VOTOOLOW. */
                                  /* vr< VREF if we are still stepping up.*/
        vr = vr + 9000;

/* Check to make sure vo is not too high. */

if (dum1 > VOTOOHI)
{
        result = 0;                                                   440
        MDACwrite();        /* Write out a zero.  */
        while(1);           /* Kill the program.  */
}

/* Steady-state action.  */

if ((dum1 < sshi) && (dum1 > sslo) && (count2 < STEADYSTATE))
{
        /* We're in the s-s band, but we haven't reached the right */
        /* number of repetitions for s-s.  */                         450

         count2 = count2 + 1;
         sum = sum + result;
 }
```

```
/* If in steady-state for STEADYSTATE cycles, set flag to 1 to indicate */
/* officially in steady-state and can fix output command based  */
/* on previous STEADYSTATE commands.                            */

if ((count2 == STEADYSTATE) && (dum1 < sshi) && (dum1 > sslo) )                460
        {
                flag = 1;
        }

else
if ((dum1 > outsshi) || (dum1 < outsslo))
        {
        /* We're not only not in steady-state, but we've stepped out */
        /* of the safety hysteresis range.  Start over. */
                                                                               470
        flag = 0;
        count2 = 0;
        sum = 0;
        }

/* Now act on the flag - send out an average command if flag is set. */
/* Otherwise start a channel 1 A/D conversion. */

if (flag == 1)
        {                                                                      480
        /* We're in steady-state, so send out the average command. */
        /* Do NOT start a channel 1 vin A/D command. */

        result = (unsigned int) (sum/STEADYSTATE);
        MDACwrite();

        }

else
        {                                                                      490
```

```
        /* Only start vin A/D conversion if not in steady-state band. */

        ad_command = 9;   /* Start channel 1.   */
        }

}       /* End of channel 0 A/D check.   */

/* See if it is channel 1. */

else if ((check & 7) == 1)                                              500
        {
        v1 = (unsigned int) ad_result_hi;
        v1 = v1 << 2;
        v2 = (unsigned int) ad_result_lo;
        v2 = v2 >> 6;
        vi = v1 + v2;

        /* Too low output - jam on the power, dude. */

        if (vo < VOTOOLOW) result = 4095;                               510
         else
                {
                /*** The actual PI control calculation. ***/

                ve = (long) vo * (long) vo;             /* Vo squared     */
                ve = vr - ve;                           /* Error          */
                res_temp = ve * h1;                     /* Gain1*error    */
                res_t = acc >> h22;
                res_t = res_t * h21;                    /* Gain2*acc      */
                res_temp = res_temp + res_t;            /* Sum            */  520
                res_t = (long) vi * (long) vi / (long)Gdig;
                                                        /* Scale Vi^2     */

                res_temp = res_temp / res_t;            /* Final calc.: */
                                                        /* Div by vi^2. */
```

```
/* Now add in the power feed foward term. */
k_temp = PowerCoef1 * vo * iin;        /* Power Gain * */
k_temp = k_temp >> PowerCoef2;         /* input power. */
      k_temp = k_temp / (res_t * (long)Gdig); /* Div. by vi^2 */                530
res_temp = res_temp + k_temp;

      /* In case of too high command. */
      if (res_temp > 4095)
               result = 4095;
        else
            {
               /* So we're operating normally at this point.   *
                * We could check for flag==0 also here,        *
                *  but channel 1 doesn't get set in motion     *                540
                * if flag==1.  It should be flag==0 already.   */


               acc_temp = acc + ve;                /* Change acc.    */

               /* Anti-windup control.  Don't let the          */
               /* accumulator grow past a certain point.       */

               if (acc_temp > ACCMAX)              /* Max accum?   */
                        acc = ACCMAX;  /* Note: ACCMAX > 0.     */              550

               else if (acc_temp < 0)              /* acc negative?*/
                        acc = 0;

               else acc = acc_temp;                /* We're okay.    */


               if (res_temp < 0)
                        result = 0;
               else                                                             560
               {
               /* Set result to the calculation answer. */
```

```
                    result = (unsigned int) res_temp;
                    }
                    }
            }

        /* Here is the big write out command. */
        /* If flag is 1, we're in steady-state, so send out
           the average.  Else send out the calculated result. */

        if (flag == 1)
                /***********************************
                 We will never get here if channel 0's
                 code doesn't trigger a channel 1 conversion when
                 flag is 1. This code is only beautific.***********/
                 {
                 result = (unsigned int) (sum/STEADYSTATE);
                 MDACwrite();
                 }
        else
        {
                /* So we have a result, and we're not in steady-state      */
                /* band.  Write out the result to the DAC.                 */
                MDACwrite();
        }

} /* End of check for channel 1.  */


/* See if it is channel 2. */

else if ((check & 7) == 2)
{
        v1 = (unsigned int) ad_result_hi;
        v1 = v1 << 2;
        v2 = (unsigned int) ad_result_lo;
        v2 = v2 >> 6;
```

```
        iin = v1 + v2;
                                                                    600
        /* Check for too high current. */

        if (iin > ITOOHI)
        {
                result = 0;
                MDACwrite();            /* Write out a zero.       */
                while(1);               /* Kill the program.       */
        }

    ad_command = 8;     /* Channel 0 */                             610

  } /* End of check for channel 2.  */

} /* End of analog_conversion_done ISR.  */

/* The main routine. */

main()
{
        /* Initialize lots of stuff. */                             620

        count1 = 0;
        count2 = 0;
        count3 = 0;
        flag = 0;
        sum = 0;
        result = 0;
        vr = 6290064;
        sslo = 2508;
        sshi = 2528;                                                630
        outsslo = 2498;
        outsshi = 2538;

        acc = 0;
```

```
        MDACinit();
        result = 0;                 /* Initial zero command.          */
        MDACwrite();            /* Write command.                 */

        ioport1 = 0;                                                    640

        int_mask = 0x22;        /* Initialize interrupts.         */
        int_pending = 0;
        hso_command = 0x18;     /* Schedule the first time ISR  */
        hso_time = timer1 + 6254;
/*      ad_command = 8; */

        enable();                   /* Enable interrupts.             */
        while(1);                   /* Let interrupts do their work.*/
                                                                        650
} /* End main */
```

E.2 Simulation MATLAB Code

As in the previous section, first the current loop single step response simulation code is presented in BASSMAN2.M. The simulation loop starts with `for i = 2:ARRAY`. Following this listing is replacement code for implementing the oscillatory and ramping current references. The final MATLAB program, BASSMAN1.M, is the voltage loop simulation. These simulations exactly match the C code.

```
% An attempt at simulating the current, voltage loop action
% in a real-time sense.
% This version implements the voltage and current loops with
% soft start and a step in current reference.
% The soft start is only until the hack voltage.
% This simulation corresponds directly to the C code BASSMAN2.C
% which runs the step response experiment
% for the full current loop.
```

```
% Define the gains for the system.
% These digital gains are scaled versions of the actual discrete
% gains.  Thus the equations are converted to containing digital
% state variables which the computer can control.

% The digital voltage loop gains.
% We will multiply by h211 and divide by h212.

h11 = 152;
h211 = 973;
h212 = 256;

% The digital current loop gains.
% We will multiply by h411 and divide by h412.

h31 = 4;
h411 = 4497;
h412 = 1024;

% The digital feedforward gain.
% We will multiply by PowerCoef1 and divide by PowerCoef2.

PowerCoef1 = 220;
PowerCoef2 = 1;

% Various defines, MATLAB style.

% Steady state count for voltage loop.
STEADYSTATE = 85;

% Steady state count for current loop.
CURSSCNT = 10;

% Offset for getting rid of resolution mapping effects.
OFFSET = 2120;
```

```
% Squared voltage reference corresponding to 169.7 Volts.
NOMVR = 2011927;

% Scaling factor.
Gdig = 2;

% Maximum digital accumulator values.
% Voltage loop max accumulator.
ACCMAX = 36762754;
% Current loop max accumulator.
CURACCMAX = 761;

% Number of voltage loop iterations before
% a current loop calculation can occur.
VOLTITER = 50;

% Soft start defines.

% The digital squared output voltage at
% the edge of the resolution mapping.
ENDSOFTSTART = 4977361;

% The final target from increasing current
% reference during closed loop part of soft start.
FINALSOFTSTART = 330;

% Values to load into the digital references
% and accumulators at the transition in soft start
% from open loop to closed loop.
HACKIR = 319;
HACKVR = 5875496;
HACKCURACC = 552;
HACKACC = 3693936;

% The number of iterations in this simulation.
ARRAY = 3000;
```

```
%% Other constants representing the physical system.

% The time in a half-line cycle.
Tl = 8.33333e-3;

% Boost converter load resistor and bus capacitor.
R = 3.91e+3;
C = 470e-6;

% A/D gain.
FAD = 204.6;

% Max rectified input voltage.
V = 168;

% Digital rectified input voltage after scaling and sampling.
V1 = floor(V*7.6e-3*FAD);

% Scaling factor for boost converter output voltage.
Divo = 9.75e-3;

% Slope of resolution mapping.
m = 4.19;

% Max input current peak to input voltage peak ratio.
kmax = .00263;

% Various arrays of state variables
% These are set up in the beginning just to speed up
% calculation.
x = zeros(1,ARRAY);
x(1) = 160;
Sv = zeros(1,ARRAY);
Sv(1) = 0;
```

```
Si = zeros(1,ARRAY);
Si(1) = 0;
k = zeros(1,ARRAY);                                                    120
k1 = zeros(1,ARRAY);
k1(1) = .000522;
d = zeros(1,ARRAY);
dd = zeros(1,ARRAY);
s = zeros(1,ARRAY);
y = zeros(1,ARRAY);
y(1) = 25600;
vo = zeros(1,ARRAY);
vo(1) = 160;
ref = zeros(1,ARRAY);                                                  130
ref(1) = HACKIR;
i1 = zeros(1,ARRAY);
id = zeros(1,ARRAY);

datavr = zeros(1,ARRAY);
datak = zeros(1,ARRAY);
datair = zeros(1,ARRAY);
datacuracc = zeros(1,ARRAY);

% First the main stuff - initialization                                140

count1 = 0;
count2 = 0;
count3 = 0;
count4 = 0;
count5 = 0;
flag = 0;
curflag = 0;
softflag = 0;
sum = 0;                                                               150
vr = NOMVR;

ir = HACKIR;
```

```
cursslo = 332;
cur… = 342;
curoutsslo = 327;
curoutsshi = 347;

acc = 0;
curacc = 0;                                                    160


% The actual loop - each iteration is the equivalent of
% one software_timer isr plus A/D conversion routines.

for i = 2:ARRAY

% Add in step response function.

if i == 1200                                                   170
        ir = 387;
end

if i == 1900
        ir = 330;
end

ref(i) = ir;

% Stuff for voltage loop control.                              180

% The power balance equation.  This tells us, based
% on the sampled large-signal Tl model what the next
% value of vo squared will be knowing the present
% vo and k.
% y represents the output voltage squared - analog.

y(i) = y(i-1) + (Tl/C)*((V*V) * k1(i) - 2*y(i-1)/R);
```

```
% The voltage is always >= recitified input voltage.                    190
if y(i) < (160*160)
        y(i) = 160*160;
end
vo(i) = sqrt(y(i));

% Now start to convert everything to digital.
% The hack produces voltages non-zero voltages starting at
% about 277 Volts.  There after there is a slope of m.
% The system does not allow voltages of over 400 Volts.
                                                                         200
if vo(i) < 277
        d(i) = 0;
elseif vo(i) > 400
        d(i) = 3.9 * m;
else d(i) = Divo * m * vo(i);
end

% Note we do not have to add in the offset.  This method
% assumes the offset is added in, so digital voltage =
% analog voltage * Divo * m * FAD.                                       210

dd(i) = floor(d(i) * FAD);
s(i) = dd(i)*dd(i);

% Now deal with soft start possibility.
if  s(i) < ENDSOFTSTART
        k(i+1) = k(i) + 3;
        k1(i+1) = kmax/4095 * k(i+1);
        softflag = 0;
        ir = HACKIR;                                                     220
        curacc = HACKCURACC;
        vr = HACKVR;
        acc = HACKACC;

else
```

```
softflag = 1;

% The error voltage is the fed back digital voltage - digital reference.

verr = floor(vr - s(i));                                                    230

% Produce a digital k based on PI control.

g = floor((floor(h11 * verr) + floor(h211*acc/h212)) / (V1*V1) * Gdig);
g = g + floor((PowerCoef1*id(i)*dd(i))/PowerCoef2/(V1*V1));
if g > 4095
        k(i+1) = 4095;
else k(i+1) = g;
end
if g < 0                                                                    240
        k(i+1) = 0;
end

% Scale the digital k so we have an analog k to plug into the power
% balance equation.

k1(i+1) = kmax/4095 * k(i+1);

% Only adjust the accumulator if the digital gain
% calculated was reasonable (within 0-4095 range).                          250

if g <= 4095
        acc1 = acc+verr;


if acc1 < 0
        acc = 0;
elseif acc1 > ACCMAX
        acc = ACCMAX;
else acc = acc1;                                                            260
end
```

```
end

% Stuff for current loop control.

% First an equation for the current.

i1(i) = vo(i)/R;

% Now convert it to digital.                                          270

id(i) = floor(i1(i) * 20 * 215);

count3 = count3 + 1;

% only do a calculation every VOLTITER iterations.
% The current loop is slower than the voltage loop,
% so only do 1 current loop calculation for every
% VOLTITER voltage loop calculations.
                                                                      280
if count3 == VOLTITER

        % Do a soft start action first of necessary.

        if softflag == 1 & (ir + 1) < FINALSOFTSTART
                ir = ir + 11;
        end

        count3 = 0;
                                                                      290
        % Now perform calculation to determine new
        % voltage reference.

        ie = floor(ir - id(i));
        vr1 = floor(floor(ie * h31) + floor(floor(curacc * h411) / h412));

        curacc1 = curacc + ie;
```

```
        if curacc1 > CURACCMAX
                curacc = CURACCMAX;
        elseif curacc1 < 0                                              300
                curacc = 0;
        else curacc = curacc1;
        end

        datacuracc(i) = curacc;

        if vr1 < 0
                vr = NOMVR;
        else vr = vr1 * vr1;
        end                                                             310

% End count3 == VOLTITER
end

% End of soft start
end

datavr(i) = vr;
datak(i) = k(i);
datair(i) = ir;                                                         320

% End of the for loop.
end
```

```
% Add in oscillatory reference function.

if i == 1200
        ir = 387;
end

if i == 1900
```

```
        ir = 330;
end

if i == 2600
        ir = 387;
end

if i == 3300
        ir = 330;
end

if i == 4000
        ir = 387;
end

if i == 4700
        ir = 330;
end

if i == 5400
        ir = 387;
end

if i == 6100
        ir = 330;
end
```

```
% Add in ramping reference function.

if i > 2600
ir = 387 - floor((i-2600)/128);
end
```

```
% An attempt at simulating the voltage loop action
```

```
% in a real-time sense.
% This version implements the voltage loop with no added cool stuff
% except a voltage reference step.
% This simulation corresponds directly to the C code BASSMAN1.C
% which runs the step response experiment
% for the voltage loop only.

% Define the gains for the system.
% These digital gains are scaled versions of the actual discrete
% gains.  Thus the equations are converted to containing digital
% state variables which the computer can control.

% The digital voltage loop gains.
% We will multiply by h211 and divide by h212.

h11 = 142;
h211 = 973;
h212 = 256;

% The digital feedforward gain.
% We will multiply by PowerCoef1 and divide by PowerCoef2.

PowerCoef1 = 220;
PowerCoef2 = 1;

% Various defines, MATLAB style.

% Steady state count for voltage loop.
STEADYSTATE = 55;

% Offset for getting rid of resolution mapping effects.
OFFSET = 2120;

% If the boost converter digital output voltage is below
% VOTOOLOW, then the max command 4095 is written.
VOTOOLOW = 2207;
```

```
% Squared voltage reference corresponding to 169.7 Volts.
NOMVR = 2011927;

% Scaling factor.
Gdig = 2;

% Maximum digital accumulator value for the voltage loop.
ACCMAX = 36762754;

% Soft start defines.

% The digital squared output voltage at
% the edge of the resolution mapping.
ENDSOFTSTART = 2324;

% The final target from increasing square voltage
% reference during closed loop part of soft start.
FINALSOFTSTART = 6290064;

% Values to load into the digital reference
% and accumulator at the transition in soft start
% from open loop to closed loop.
HACKVOLTACC = 3693936;
HACKVR = 5875496;

% The number of iterations in this simulation.
ARRAY = 2500;

%% Other constants representing the physical system.

% The time in a half-line cycle.
Tl = 8.33333e-3;

% Boost converter load resistor and bus capacitor.
R = 3.91e3;
```

```
C = 470e-6;

% A/D gain.
FAD = 204.6;

% Max rectified input voltage.
V = 168;

% Digital rectified input voltage after scaling and sampling.
V1 = floor(V*7.6e-3*FAD);

% Scaling factor for boost converter output voltage.
Divo = 9.75e-3;

% Slope of resolution mapping.
m = 4.19;

% Max input current peak to input voltage peak ratio.
kmax = .00263;


% Various arrays of state variables
% These are set up in the beginning just to speed up
% calculation.
x = zeros(1,ARRAY);
x(1) = 160;
Sv = zeros(1,ARRAY);
Sv(1) = 0;
Si = zeros(1,ARRAY);
Si(1) = 0;
k = zeros(1,ARRAY);
k1 = zeros(1,ARRAY);
k1(1) = .000522;
d = zeros(1,ARRAY);
dd = zeros(1,ARRAY);
s = zeros(1,ARRAY);
```

```
y = zeros(1,ARRAY);
y(1) = 25600;
vo = zeros(1,ARRAY);
vo(1) = 160;
ref = zeros(1,ARRAY);
ref(1) = HACKVR;
i1 = zeros(1,ARRAY);
i1(1) = vo(1)/R;
id = zeros(1,ARRAY);
id(1) = floor(i1(1) * 20 * 215);

% First the main stuff - initialization

count1 = 0;
count2 = 0;
count3 = 0;
count4 = 0;
count5 = 0;
flag = 0;
curflag = 0;
sum = 0;
result = 0;

vr = 6290064;
acc = 0;


% The actual loop - each iteration is the equivalent of
% one software_timer isr plus A/D conversion routines.

for i = 2:ARRAY

% Change the voltage reference.

if i == 1200
```

```
vr = 9054081;
%vr = 8854184;
end

if i == 1900                                                    150
vr = 6290064;
%vr = 5946800;
end

ref(i) = vr;

% The power balance equation.  This tells us, based
% on the sampled large-signal Tl model what the next
% value of vo squared will be knowing the present
% vo and k.                                                     160
% y represents the output voltage squared - analog.

y(i) = y(i-1) + (Tl/C)*((V*V) * k1(i) - 2*y(i-1)/R);

% The voltage is always >= recitified input voltage.
if y(i) < (160*160)
        y(i) = 160*160;
end
vo(i) = sqrt(y(i));
                                                                170
% Now start to convert everything to digital.
% The hack produces voltages non-zero voltages starting at
% about 277 Volts.  There after there is a slope of m.
% The system does not allow voltages of over 400 Volts.

if vo(i) < 277
        d(i) = 0;
elseif vo(i) > 400
        d(i) = 3.9 * m;
else d(i) = Divo * m * vo(i);                                   180
end
```

```
% Calculate the analog and digital current.

i1(i) = vo(i) / R;
id(i) = floor(i1(i) * 20 * 215);

% Note we do not have to add in the offset.  This method
% assumes the offset is added in, so digital voltage =
% analog voltage * Divo * m * FAD (=ADC gain).                    190

dd(i) = floor(d(i) * FAD);
s(i) = dd(i)*dd(i);

% Now deal with soft start possibility.
if  dd(i) < ENDSOFTSTART
        k(i+1) = k(i) + 3;
        k1(i+1) = kmax/4095 * k(i+1);
        softflag = 0;
        vr = HACKVR;                                              200
        acc = HACKVOLTACC;
else
softflag = 1;

if softflag == 1 & vr < FINALSOFTSTART
        vr = vr + 9000;
end

% First see if vo is too low.  Too low output -
% jam on the power, dude.                                         210

if dd(i) < VOTOOLOW
        k(i+1) = 4095;
else

% The error voltage is the fed back digital voltage - digital reference.
```

```
verr = floor(vr - s(i));

% Produce a digital k based on PI control.                                    220

g = floor( Gdig*(floor(h11 * verr) + floor(h211*acc/h212)) / (V1*V1));
g = g + floor((PowerCoef1*id(i)*dd(i))/PowerCoef2/(V1*V1));
if g > 4095
        k(i+1) = 4095;
else k(i+1) = g;
end
if g < 0
        k(i+1) = 0;
end                                                                          230

end

% Scale the digital k so we have an analog k to plug into the power
% balance equation.

k1(i+1) = kmax/4095 * k(i+1);

% Only adjust the accumulator if the digital command
% calculated was reasonable (within 0-4095 range).                           240

 if g <= 4095
        acc1 = acc+verr;


if acc1 < 0
        acc = 0;
elseif acc1 > ACCMAX
        acc = ACCMAX;
else acc = acc1;                                                             250
end
end
end
```

```
% End of the for loop.
end
```

E.3 Low Pass Filter MATLAB Code

Both boost converter output voltage and output current sensing circuits contain low pass filters. This MATLAB program constructs the Bode plots.

```
%
% Simple program for entering
% a three RC-RC stage low pass filter
% and analyzing its frequency
% response.
%

R1 = 280;
R2 = 2.25e+3;
C1 = 1e-6;
C2 = 3.3e-6;

num1=1;
num2=1;
num3=1;

den1 = [R1*R2*C1*C2 (R1*C2 + R2*C2 + R1*C1) 1];
den2 = den1;
den3 = den1;

num = conv(conv(num1,num2),num3);
den = conv(conv(den1,den2),den3);
bode(num,den)
```

E.4 Step Invariant Transform MATLAB Code

This MATLAB program calculates and plots the continuous and discrete step responses of the CT and DT models of the circuit in Fig. 2-9. The step invariant transform was used to generate the discrete-time model.

```
% Simulation to evaluate step invariant transform.
% First we will simulate the continuous time response.

R1 = 3.91e+3;
R2 = 50;
C = 10e-3;
T = 1/120;
ToverTAU = T/(R2*C);

contnum = [C*(1 + R2/R1) 1/R1];
contden = [C*R2 1];

% A step response input.

Uc = [ones(360,1)];

% Create a time vector.

ctime = ones(1:360);
for i = 2:360
  ctime(i) = ctime(i-1) + 1;
end
ctime = ctime * T;

% Perform the simulation for continuous time.

[yc, xc] = lsim(contnum, contden, Uc, ctime);


% Now input the digital system.
```

```
% We have the digital transfer function G(z)

dignum = [(1/R1 + 1/R2) -(exp(-ToverTAU)/R1 + 1/R2)];
digden = [1 -(exp(-ToverTAU))];

% A step response input.

input = [ones(360,1)];

% Create a time vector.                                          40

dtime = ones(1:360);
for i = 2:360
  dtime(i) = dtime(i-1) + 1;
end
dtime = dtime * T;

% Perform the simulation for discrete time.

[yd,xd] = dlsim(dignum, digden, input);                          50

% So at this point we have two simulations, both
% with manually inputted step responses.
% Each response lasts for three seconds or so.
% First we will normalize them to a max initial
% value of 1.  Then we will plot both of them simultaneously.

yd = yd/yd(1);
yc = yc/yc(1);
                                                                 60
plot(ctime, yc, '--', dtime, yd, ':')

xlabel('Time (seconds)')

ylabel('Normalized current')
```

E.5 Uploading Scope Pictures

The first program, GETWFM.BAS, uploads pictures from a Tektronix TDS320 oscilloscope to a PC. The program CHANGEFILES.C converts the waveform file into a MATLAB loadable file.

```
'
' getwfm.bas – program to get a waveform from an oscilloscope,
'              convert its data values to absolute voltage measurements,
'          and store these values in a Lotus 123 compatible file.
'
' Version 1.2.
'

'Global Status Variables
COMMON SHARED COUNT%   'Number of characters read or written                    10

'I/O Subroutines
DECLARE SUB RS232WRITE (file%, stringVar$)
DECLARE SUB RS232READ (file%, stringVar$)
DECLARE SUB RS232WAITCOM (file%, delay%)
DECLARE SUB SENDBREAK (file%, port%)

'
' rs232io.bas – collection of input/output routines to be used by the
'               example programs                                                 20
'



'TIMEOUT:
'  PRINT "Timeout"
'  CLOSE #file%
' END


'Timeout:                                                                        30
```

```
'   PRINT "Timeout"
'     CLOSE #file%
'   END



'
' Allocate the buffers and initialize the port for input and output.
'  Avoid input buffer overflow on slow machines by setting buffer size
' to 1000 bytes.  Assign a unique identifier to the port and store                    40
'  in variable SCOPE.
'
   CLS
   PRINT : PRINT "         TDS 300 Series GETWFM Program - Version 1.2"
   PRINT : PRINT
   PRINT "    ***  RS-232 parameters should be set to default values:"
   PRINT
   PRINT "           Baud Rate:  9600      EOL: LF        Delay:  0 s"
   PRINT "           Hard Flagging:  On    Parity:  None"
   PRINT "           Soft Flagging:  Off   Stop Bits:  1"                           50
   PRINT
   PRINT "         These parameters can be changed in the "
   PRINT "         UTILITY  System - I/O  RS-232 menu."
   PRINT
   DO
        INPUT "    ***  Press (1) to use COM1 or (2) for COM2 >", port$
   LOOP UNTIL (port$ = "1" OR port$ = "2")

   SCOPE% = 1
   OPEN "COM" + port$ + ":9600,N,8,1,RB1000" FOR RANDOM AS #SCOPE%                  60
'
' Clear the device and check for errors
'
    CALL SENDBREAK(SCOPE%, VAL(port$))
'
'  Turn off the header from query responses.
```

```
'
        CALL RS232WRITE(SCOPE%, "HEADER OFF")
'
' Set up the data source to be channel 1.                                    70
'
        DATSRC$ = "ref1"
        WRT$ = "DATA:SOURCE " + DATSRC$
        CALL RS232WRITE(SCOPE%, WRT$)
'
'  Set up data encoding to be ribinary and data width to 1.
'
        CALL RS232WRITE(SCOPE%, "DATA:ENCDG RIBINARY;WIDTH 1")
'
' Print a message on the screen instructing the user to connect Channel 1    80
'  to the test signal.
'
        CLS
        PRINT : PRINT "         TDS 300 Series GETWFM Program - Version 1.2"
        PRINT : PRINT
        PRINT "    ***  Connect your test signal to Channel 1  ***"
        PRINT
        PRINT "         For example, you might connect the PROBE COMP signal"
        PRINT "         to the Channel 1 input connector using a 10X probe."
        PRINT                                                               90
        PRINT "    ***  Press any key when done  ";
        ANS$ = INPUT$(1)
'
' Set up the recordlength, and the data start and data stop positions.
'  In this example, the entire waveform is obtained so data start and
' data stop are 1 and 1000 respectively.
'
        RL% = 1000
        WRT$ = "HORIZONTAL:RECORDLENGTH " + STR$(RL%)
        CALL RS232WRITE(SCOPE%, WRT$)                                      100

        DSTART% = 1
```

```
        DSTOP% = RL%
        WFPOINTS% = DSTOP% - DSTART% + 1
        PRINT : PRINT : PRINT
        PRINT "        Number of waveform points is "; WFPOINTS%
        PRINT

        WRT$ = "DATA:START " + STR$(DSTART%)
        CALL RS232WRITE(SCOPE%, WRT$)                                    110

        WRT$ = "DATA:STOP " + STR$(DSTOP%)
        CALL RS232WRITE(SCOPE%, WRT$)

        WRT$ = "HEADER OFF"
        CALL RS232WRITE(SCOPE%, WRT$)
'
'  Make sure setup changes have taken effect and a new waveform is acquired
'
        CALL RS232WRITE(SCOPE%, "ACQUIRE:STATE RUN")                     120
'
'  Wait for the scope to acquire the waveform.
'
        CALL RS232WAITCOM(SCOPE%, 10)
'
'  Send the scope a curve query to get waveform data.
'
        CALL RS232WRITE(SCOPE%, "CURVE?")
'
'  Read waveform data into a string.                                     130
'
        CALL RS232READ(SCOPE%, WFM$)
'
'  Transfer waveform data into an array.    There is one data point
'  stored in each array element.
'
        DIM WFARR%(1 TO WFPOINTS%)      ' set up the waveform output array
        FOR I% = 1 TO WFPOINTS%
```

```
        WFARR%(I%) = ASC(MID$(WFM$, I%, 1))
        IF WFARR%(I%) > 127 THEN WFARR%(I%) = WFARR%(I%) - 256
      NEXT I%
'
' Read the waveform preamble.
'  Get the vertical offset and scale multiplier,
' the trigger point, the horizontal sampling interval
'  and the horizontal units to convert the data points to
' time and voltage values.
'
      WRT$ = "WFMPRE:" + DATSRC$ + ":YOFF?"
      CALL RS232WRITE(SCOPE%, WRT$)
      CALL RS232READ(SCOPE%, rd$)
      YOFF! = VAL(rd$)

      WRT$ = "WFMPRE:" + DATSRC$ + ":YMULT?"
      CALL RS232WRITE(SCOPE%, WRT$)
      CALL RS232READ(SCOPE%, rd$)
      YMULT! = VAL(rd$)

      WRT$ = "WFMPRE:" + DATSRC$ + ":YUNIT?"
      CALL RS232WRITE(SCOPE%, WRT$)
      CALL RS232READ(SCOPE%, YUNIT$)

      WRT$ = "WFMPRE:" + DATSRC$ + ":PT_Off?"
      CALL RS232WRITE(SCOPE%, WRT$)
      CALL RS232READ(SCOPE%, rd$)
      PtOff!  = VAL(rd$)

      WRT$ = "WFMPRE:" + DATSRC$ + ":XINCR?"
      CALL RS232WRITE(SCOPE%, WRT$)
      CALL RS232READ(SCOPE%, rd$)
      XINCR! = VAL(rd$)

      WRT$ = "WFMPRE:" + DATSRC$ + ":XUNIT?"
      CALL RS232WRITE(SCOPE%, WRT$)
```

```
        CALL RS232READ(SCOPE%, XUNIT$)
'
'       Output header (x-, y-units, date, time and source)
'     Write out the data to a file called "WFMXXXX.PRN".
'
        WAVEFL$ = "WFMXXXX.PRN"          ' set up an output file for the data time and voltages   180
        OPEN WAVEFL$ FOR OUTPUT AS #2
        PRINT #2, XUNIT$; ","; YUNIT$
        WRITE #2, DATE$, TIME$, "ref1"
'
'     Process waveform data.
'       Write out the data to a file called "WFMXXXX.PRN".
'
        DIM VOLTARR!(1 TO WFPOINTS%) ' set up the voltage output array
        DIM TIMEARR!(1 TO WFPOINTS%) ' set up the time output array
        FOR I% = 1 TO WFPOINTS%                                                          190
                TIMEARR!(I%) = (I% - 1 - PtOff!) * XINCR!
                VOLTARR!(I%) = (WFARR%(I%) - YOFF!) * YMULT!
                PRINT #2, TIMEARR!(I%), VOLTARR!(I%)
        NEXT I%
'
'     Final message
'
        PRINT "        Waveform time and voltage points are in file ";
        PRINT WAVEFL$
        PRINT                                                                            200
'
'  Deallocate buffers and close communications.
'
        CLOSE #SCOPE%
        END

'
' RS232READ - reads strings and binary and ASCII blocks into a string from
'  RS-232.    For strings, reads until CR or LF terminator is seen.    If a block
' is detected, the length of block is read and a terminator is scanned for.        210
```

```
'  CR/LF and LF/CR are not valid terminators.
'
SUB RS232READ (file%, stringVar$)
   COUNT% = 0        'reset byte count
    stringVar$ = ""      'clear string

   lengthToRead% = 1   'always 1 for strings
    timeoutSec% = 5      'set timeout for first byte for 5 sec

   'Block status variables                                                    220
    bin% = 0             'Boolean value: 0 = read string, 1 = read block
   blockState% = 0

   DO
'       ON TIMER(timeoutSec%) gosub TIMEOUT
       TIMER ON

       WHILE (EOF(1))    'loop until char in buffer or timeout
       WEND
                                                                             230
       TIMER OFF       'char received

       '
       '  Get character(s):  for blocks, read length of block or what is
       ' available in the buffer; for strings read 1 char
       '
       lengthInBuffer% = LOC(1)
       IF (lengthToRead% > lengthInBuffer%) THEN
           ch$ = INPUT$(lengthInBuffer%, #file%)
           lengthRead% = lengthInBuffer%                                     240
           COUNT% = COUNT% + lengthRead%
       ELSE
           ch$ = INPUT$(lengthToRead%, #file%)
           lengthRead% = lengthToRead%
           COUNT% = COUNT% + lengthRead%
       END IF
```

```
timeoutSec% = 1      'first byte received; wait up to 1 sec for
                    'following chars.

'                                                                    250
'  Blocks are formatted as #<x><yyy><data><newline> where
'     <x> is the number of y bytes; for example if yyy = 500, then
'               x = 3
'     <yyy> is the number of bytes to transfer including checksum;
'               if width is 1 then all bytes on bus are single data
'          points; if width is 2 then bytes on bus are
'               2-byte pairs; this program uses width of 1
'     <data> is the curve data
'        <newline> is a single byte newline character at the end of
'          the data                                                   260
'
' State machine for interpreting binary and ASCII blocks:
'
'  blockState%
'        0         Initial state; remains in this state until # char
'              is received.
'        1         Read <x>
'     2      Read <yyy>
'        3         Read length of block
'     4      Scan for a CR or LF terminator then exit subroutine     270
'
IF (bin%) THEN
    SELECT CASE blockState%
       CASE 1
           IF ((ch$ >= "1") AND (ch$ <= "9")) THEN
               blockState% = 2
               nzdig% = VAL(ch$)
               blockSize% = 0
           ELSE
               blockState% = 0                                       280
               bin% = 0
           END IF
```

```
        CASE 2
            IF ((ch$ >= "0") AND (ch$ <= "9")) THEN
                blockSize% = (blockSize% * 10) + VAL(ch$)
                nzdig% = nzdig% - 1
                IF (nzdig% = 0) THEN
                    lengthToRead% = blockSize%    'does not include
                  blockState% = 3                 'terminator
                END IF                                                    290
            ELSE
                blockState% = 0
                bin% = 0
            END IF
        CASE 3
            lengthToRead% = lengthToRead% - lengthRead%
            IF (lengthToRead% = 0) THEN
                blockState% = 4
                lengthToRead% = 1    'scan for terminator
          END IF                                                          300
                stringVar$ = stringVar$ + ch$
        CASE 4
          IF ((ch$ = CHR$(10)) OR (ch$ = CHR$(13))) THEN
                EXIT DO
          END IF
    END SELECT
ELSE
    SELECT CASE ch$
        CASE CHR$(10), CHR$(13)       'scan for terminator
            EXIT DO                                                       310
        CASE "#"                          'block detected
          IF (blockState% = 0) THEN
                bin% = 1
                blockState% = 1
          END IF
        CASE ELSE
          stringVar$ = stringVar$ + ch$
    END SELECT
```

```
        END IF
    LOOP                                                              320
END SUB

'
' RS232WAITCOM – wait for a command to finish by doing a *OPC? query
'  and reading its results; wait only as long as the delay value.
'
SUB RS232WAITCOM (file%, delay%)
'     ON TIMER(delay%) gosub TIMEOUT
      TIMER ON
      CALL RS232WRITE(file%, "*OPC?")      '*OPC? places a 1 in the Output   330
    CALL RS232READ(file%, rd$)        'Queue once an operation is complete
      TIMER OFF
END SUB

'
'  RS232WRITE - send the contents of the string to the device and wait
' for the write to finish.
'
SUB RS232WRITE (file%, stringVar$)
      COUNT% = 0        'reset byte count                               340

    bufferSize% = LOF(1)
    PRINT #file%, stringVar$ + CHR$(13)   'send command + CR terminator

'     ON TIMER(5) gosub TIMEOUT       'set 5 sec timeout for write to finish
      TIMER ON

      WHILE (LOF(1) < bufferSize%)      'loop until all chars in buffer are
    WEND                        'sent or timeout
                                                                        350
      TIMER OFF                         'chars sent
    COUNT% = LEN(stringVar$) + 1
END SUB
```

```
'
' SENDBREAK – sends break over RS–232 and reads DCL response from
'  instrument.    Timeout (in RS232READ) will occur if no response.
'
SUB SENDBREAK (file%, port%)
   CONST COM1 = &H3FB  ' Standard addresses for Line Control Register
    CONST COM2 = &H2FB   ' Changes may be necessary for machines with
                        ' unconventional addresses.

    IF (port% = 2) THEN comm% = COM2 ELSE comm% = COM1

    lcr% = INP(comm%)                'save register state

   OUT comm%, (lcr% OR 64)     'set break bit (bit 6 in LCR)
    SLEEP (1)                        'wait 1 s for register update
   OUT comm%, lcr%             'restore register state
    CALL RS232READ(file%, message$)   'read for DCL message; receive or timeout
END SUB
```

```
/* This program converts a .PRN file from GETWFM.BAS
* uploading of TDS 320 series Tektronix scope to an
* ascii MATLAB file.  The output text file contains
* a column of times and a matching column of voltages.
*/

/*
* Simply type in the number of the .PRN file.  The program
* assumes a WFMXXXX.PRN file exists, and creates
* a WFMXXXX.MAT file.  Entering '-1' will halt
* the program.
*/

#include <stdio.h>
```

```
FILE *fpin, *fpout;

main()
{
  char number[20];                                                    20
  char filename[20];
  char output[20];
  char tempstring[100];

  while(1)
   {

     printf("Another number WFM*.PRN file:  ");
     gets(number);
     printf("\n");                                                    30

     if (!(strcmp("-1",number)))
          exit(0);

     /* Make an input file name WFMXXXX.PRN from input XXXX. */
     /* Make an output file name WFMXXXX.MAT from input XXXX. */

     strcpy(filename,"wfm");
     strcat(filename,number);
     strcpy(output,filename);                                         40
     strcat(output,".mat");
     strcat(filename,".prn");

     if ((fpin = fopen(filename,"r")) == 0)
          {
           printf("File not found.\n");
           continue;
          }
     fpout = fopen(output,"w");
                                                                      50
     /* Read in the first two lines of junk. */
```

```
    fgets(tempstring,10000,fpin);
    fgets(tempstring,10000,fpin);

    /* Read in a line, then print it to the output file. */

    while(!(feof(fpin)))
        {
         fgets(tempstring,10000,fpin);                                    60
         fputs(tempstring,fpout);
        }

    fclose(fpin);
    fclose(fpout);
  }


}
```

E.6 EPROM File

A 70 % duty cycle switching scheme is encoded in the following EPROM file.

$\#SET\_ADDRESS = 0;$

```
11 11 11 11 11 11 11
11 11 11 11 11 11 11
11 11 11 11 11 11 11
11 11 11 11 11 11 11
11 11 11 11 11 11 11
11 11 11 11 11 11 11
11 11 11 11 11 11 11
11 11 11 11 11 11 11
11 11 11 11 11 11 11
```

```
10 10 10 10 10 10 10
10 10 10 10 10 10 10
10 10 10 10 10 10 10
10 10 10 10 10 10 10
10 10 12 12 12 12 12
12 12 12 12 12 12 12
12 12 12 12 12 12 12
12 12 12 12 12 12 12
12 12 12 12 12 12 12
12 12 12 12 12 12 12
12 12 12 12 12 12 12
12 12 12 12 12 12 12
12 12 12 12 12 12 12
12 12 10 10 10 10 10
10 10 10 10 10 10 10
10 10 10 10 10 10 10
10 10 10 10 10 10 10
10 10 10 10 00 f0 f0
f0 f0 f0 f0 f0 f0 f0
.  .  .
.  .  .
.  .  .
```

Appendix F

Capacitive Coupling System

This appendix contains the paper, written by Steven Shaw, that explains the capacitive coupling system which Mr. Shaw designed [43]. This system forms the interface between the information at the battery, such as output voltage and output current, and the charging current controller.

F.1 Background

A high frequency switching power supply can be made with a relatively compact transformer, opening the possibility of a contactless power connector. For example, the magnetic material may be placed in the joint of a robotic arm, transmitting power without wires or brushes. Alternatively, the magnetic structure might replace the contacts of conventional power connectors, providing power transmission without the problems of electrical contacts. An interesting problem is how to transmit data, for purposes of control and monitoring, through the same connector as the power without resorting to electrical contacts. The project pursued in this laboratory addresses this problem of communication and ultimately resulted in the design, printed circuit fabrication and testing of a 5 Mbit/s digital link. Overall, the system consists of two printed circuit boards, one for transmit and one for receive, and two identical annular capacitor plates forming the link between the transmit and receive circuits.

F.2 Design of Capacitive Coupling

At the distance of separation determined by the magnetic material geometry and the frequencies used in the transmit and receive boards, the electromagnetic mechanism is quasistatic. Therefore, design of the coupling structure could have taken one or a combination of two approaches; inductive or capacitive. Capacitive coupling was chosen because it seemed that leakage flux from the power magnetics would have a less pronounced effect on the transmitted signal. The inductive design would have involved a loop of copper instead of a sheet of copper, and any time rate of change in the leakage flux through the loop would clearly result in a noise voltage that the receiver circuit would have to reject. The only effect of flux leakage on the capacitive design would be to induce eddy currents in the copper plate. Instead of influencing the signal, these eddy current would cause Joule heating in the copper. The capacitor plates (Figure F-1) were constructed by placing ordinary epoxy fiberglass 1 oz copper coated material in a lathe and turning away excess copper. For purposes of experimentation the plates were supported on a pedestal and separated by roughly .06" to .1". Arrangements for positioning the plates were deliberately sloppy, simulating the misuse that a connector might receive in the field. The capacitor plates were dimensioned for the magnetic structure to be used in the electric car power converter, pending completion of the power supply design.

F.3 Design of Transmitter/Receiver Circuits

The original design for the transmitter and receiver circuits involved the frequency shift keying (FSK) of a digital signal and the modulation/demodulation of this digital signal to a frequency range where the power electronics interference would be minimized. This sort of modulation, which is essentially FM, was hoped to have the noise rejection characteristics typical of FM radio. However, getting a megahertz phase locked loop demodulator to work under the time constraints of the project without resorting to linear ASIC's designed for commercial radio frequencies seemed improbable. An alternate modulation scheme was developed that would not only have noise immunity, but would also transmit the clock

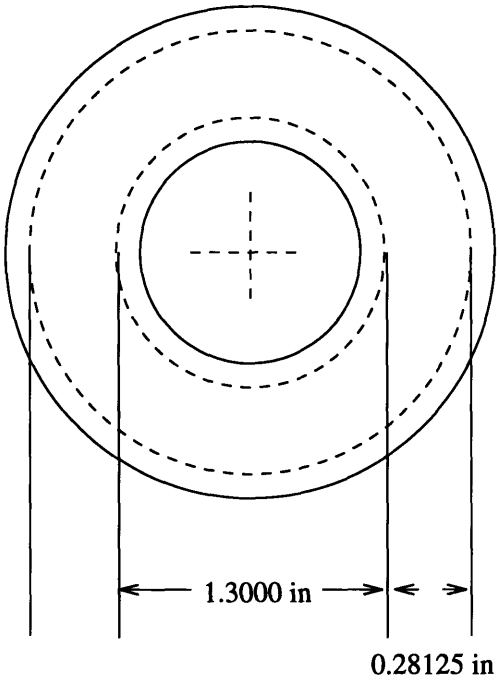

Figure F-1: Capacitor plate.

and data in a easily recoverable form. Essentially, discrete AM modulation was adapted to the purposes of the project (Figure F-2)). The advantages of this form of modulation are that modulation and demodulation is easily accomplished and the amount of analog signal processing is minimal. The circuit used to transmit the signal is extremely elegant and simple, with no linear modulators or complex passive networks (Figure 3). The stream of digital information is converted to the analog (modulated) signal to be transmitted by a three bit digital to analog (D/A) converter constructed from a programmable logic device (U5), a weighted resistor network (R1,R2,R3,R13), three transistors (Q1, Q2, Q3) and an LT1191 high frequency operational amplifier (U3). A three bit D/A converter was specified to increase flexibility; the AM modulation only uses four discrete levels, hence a two bit converter would have sufficed. The transistors act as switches, and the relative weighting of the D/A bits is accomplished by the resistors going to the inverting terminal of the op amp (U5). The amplitude modulation is done digitally by the programmable logic device (PLD) – the only analog component is the op amp, which sums the binary weighted currents and drives the capacitor plate. The extra resistor is present to bias the op amp so the center of scale on the output is zero volts.

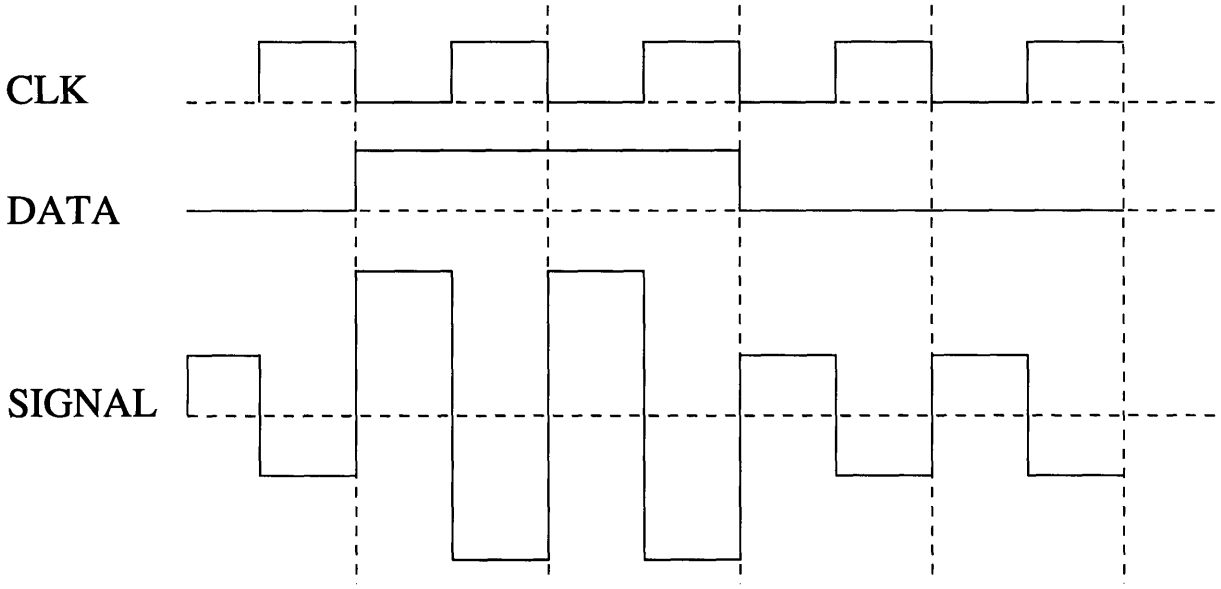

Figure F-2: AM modulation.

Decoding of the amplitude modulated signal could be accomplished by any of a variety of non linear transformations. For example, x(t)2n , n 1,2,3 ... would recover the data and tend to shrink signals with a magnitude less than one towards zero and cause larger signals to blow up. The most obvious way to recover the clock signal is just to amplify the input and clip it, using, for example, a comparator. The transfer characteristic chosen for the recovery of data is illustrated in Figure F-3. This transfer characteristic was selected because it can be easily realized by exploiting diode drops (Figure 5). The decoding circuitry works as follows. The input is a basic inverting gain stage, using another LT1191 high frequency op amp (U11). The input to this stage is connected directly to the capacitor plate. The gain setting resistors are chosen to fix an appropriate gain and also to make the loading of the capacitor plate appear to be 1K ohm, which was found to work well in the lab. The amplified input is then routed to two comparators, both of which are LT1016 "Ultra Fast Precision Comparators." The LT1016 is crucial to the operation of the receive board; it provides a 50 GHz gain- bandwidth product and a 10ns propagation delay for a 5 mV overdrive, yet is free of oscillations and other comparator pitfalls. One LT1016 (U7) simply acts as an open loop, high gain stage to derive the clock from the input signal. The other comparator (U2) has a diode drop discriminator circuit designed to extract the digital information. This comparator (U2) is wired so that if the input is less than .6 V peak to peak, signifying a binary "0", the

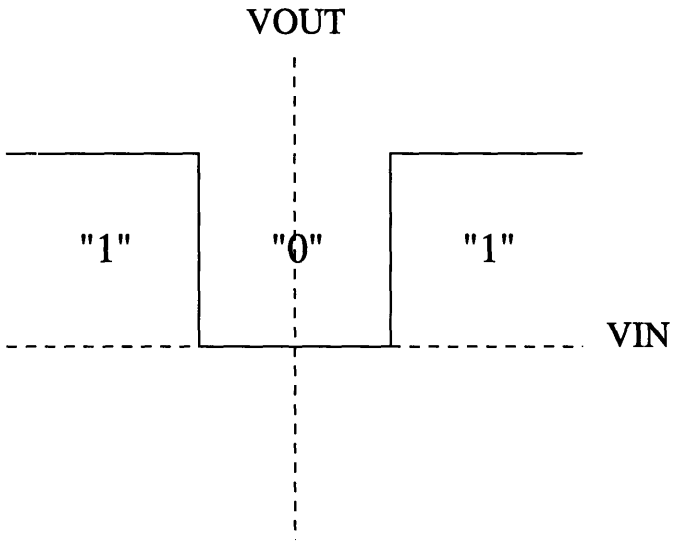

Figure F-3: Demodulator transfer diagram.

output will be zero. Stability in the "0" state is ensured by the large resistor trickling current into the inverting input when both diodes are not conducting. When the signal is larger, signifying a binary "1", the diodes go into conduction and cause the comparator to output a "1". The comparator outputs, clock and data, are then input to the "receive" digital logic.

There are a number of points to be made about the diode drop demodulator. The first point is that this modulation scheme is a poor choice if there is much variation in signal strength, as in radio. This was not deemed serious because at the operating frequency of the transmitter the variation in spacing of the capacitor plates would not change the signal strength significantly. The second point is that the AM detector is capable of decoding a signal where the information frequency is up to half of the carrier frequency; with a conventional diode-RC envelope detector, the ratio of carrier to modulation frequency would be closer to ten to one. The third point to notice is that there is the potential for a pathology – since the valid input ranges for a "1" output lie on either side of the transfer diagram, there is the possibility for invalid behavior during the zero crossing when a "1" is being transmitted (Figure F-4). From the analog standpoint, such a pathology is no real concern; it simply requires the insertion of delay along either clock or data signal paths as appropriate until the setup and hold requirements of the flip flop in the PLD (U8) are met. In the laboratory, although the comparator was observed to be fast enough to demonstrate the "invalid" behavior, setup and hold requirements were met without any need to deliberately

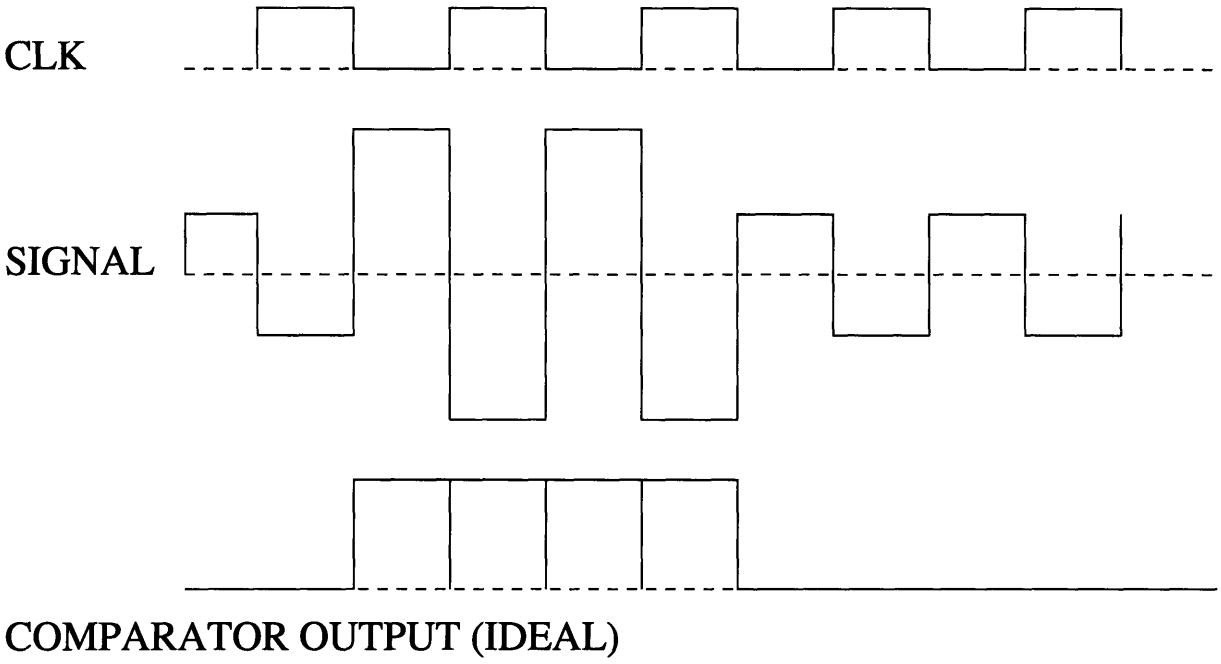

Figure F-4: Static hazards in demodulation.

insert delays. Figure 7 shows the results of the diode discriminator circuit from a scope photo. The uppermost trace shows the output of the input gain stage; this is just a slightly amplified version of what the receiving capacitor plate picks up. The transmitted signal, in this case, was just a string ...1100110011... at 5 Mbits/s; notice how each high and low level is 400ns wide. The "data" value from this input is shown correctly decoded on the middle trace. This data value is not the actual comparator output, but the comparator output as clocked into a flip flop in the PLD. The "clock" is shown on the bottom trace; this is the 10 MHz output of the clock generating comparator that drives the receive finite state machine.

F.4 Design of the Finite State Machines

The use of PLDs greatly simplified initial design by postponing the specifics of the operation of the transmitter/receiver boards until finished printed circuit boards were available for testing in the laboratory. Unfortunately, however, the design of the transmit and receive state machines was complicated by the programmable logic devices used, GAL22V10's, due to limitations in state and number of product terms. A few specifics should help clarify

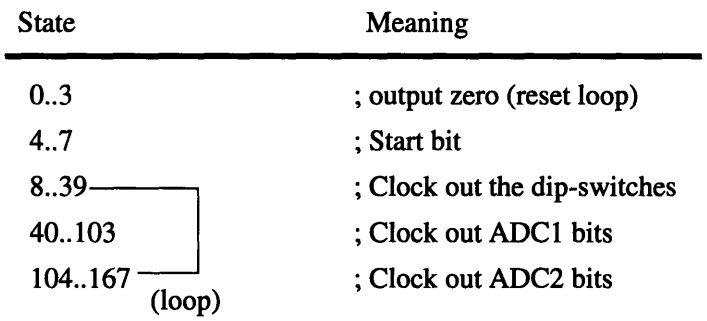

Figure F-5: Abbreviated transmitter state machine.

the design of the state machines. The transmit board operates at 20 MHz, and requires four clocks to send a bit of data, yielding an effective bit rate of 5 Mbits/s. The receive board, since it derives its clock from the input signal, runs at a clock frequency of 10 MHz; therefore there are only two clock cycles per bit of data on the receive side. The transmit state machine is required to control and read data from a bank of eight dip-switches and two MAX176 12 bit serial A/D converters, and transmit this data to the receive board. The receive state machine is required to synchronize itself with the transmit state machine and distribute the transmitted data to a bank of light emitting diodes (LEDs) that reflect the dip-switch settings, and a MAX527 Quad 12 bit D/A converter that reconstructs the analog inputs to the A/D converters on the transmit board. The receive state machine uses two eight bit shift registers constructed from 74HCT574 octal registers to convert serial data to byte oriented data. One shift register actually drives the LED's, while the other is connected to the MAX527.

The transmit board makes use of two programmable logic devices. The third device (U1) shown in the circuit diagram (see appendix) was included for flexibility but not used in the final design. One PLD, described in the file TX1.PLD in the appendix, is simply an eight bit state machine resembling a counter, while the other PLD (TX2.PLD) translates the bits of state to the functions required to control the two analog to digital converters and the output circuitry. TX1.PLD also contains the output F, which is the bit of data to be transmitted. Depending on the state, F is selected from either the dipswitches or the A/D converters. Figure F-5 is an abbreviated state diagram of the transmitter state machine, TX1.PLD.

State	Meaning
0	; Wait until 1 is received
1	
2..17	; Load lights
18..49	; Load DAC#1
50..81	; Load DAC#2

Figure F-6: Abbreviated transmitter state machine.

The lower two bits of state are used to encode the transmitted data. Two full oscillations of either low or high magnitude are transmitted during these states. The transmit FSM waits in the 0..3 state, transmitting a zero, until the reset line goes high. The power-on reset time constants for the transmit and receive boards are arranged so that the receive board is waiting for the start bit long before the start bit is transmitted. If the receive and transmit boards are operated off of independent power supplies, the receive board must be synchronized by resetting both boards manually and releasing the transmit reset button last. In state 4..7, a one (start bit) is transmitted. In the next 32 states, states 8..39, the dipswitches are scanned by the state machine and transmitted. Recall that each bit requires four states of the 20 Mhz clock to be transmitted, so the eight dipswitches use 32 states. In next 128 states (40..167) the A/D converter data is transmitted, and the state machine loops back to the transmission of the dipswitches.

The receive state machine shown in Figure F-6 is similar to the transmit state machine, but with fewer states. Synchronization of the receive and transmit state machines is accomplished by waiting for a start bit. Recall that the receive state machine has half as many states as the transmit state machine because it operates at half the clock frequency.

The PLD code RX1.PLD (see appendix) implements the state machine above in the CUPL language. The PLD code in RX2.PLD consists of expressions that drive the various control lines of the MAX527 D/A converter and the shift registers used to convert the incoming serial data to a parallel representation. For example, to drive the LEDs, the expression CLKLIGHTS.D in RX2.PLD outputs a series of clock pulses during states 2..17 which load the data intended for the light display into the LED shift register. Similarly,

CLKDA.D is programmed with an expression that clocks a shift register connected to the MAX527. There is an unused output in RX2.PLD which could be used to implement a flag for an all-digital link.

F.5 Conclusions

The circuits designed and built for this laboratory worked as expected. Figure 8 shows the simultaneous transmission of two signals from two locked signal generators. The circuits, as configured, are capable of transmitting two analog signals at a sample rate of about 128 kS/s and roughly 128 kBytes/s of digital data through the LED and dip-switch ports. By simple reprogramming of the PLDs, and with no hardware modifications, the same circuits are capable of 5 Mbits/s digital only communications or rough 200 kHz analog single channel and 200 kBytes/s digital data transfer. This performance is truly "high-bandwidth" for a simple connectorless system, and more than meets the design goals set forth at the beginning of the project. The versatility of the digital transmission scheme is an additional bonus. Desirable refinements and additions to the system include the used of surface mount components in the analog sections of the board, increasing the basic clock frequency to 40 Mhz, and using logic devices with more available state. The last improvement is probably the most necessary, as considerable effort was expended to get the PLD programs "small" enough to the work in the GAL22V10's. Reliability and robustness, although observed to be acceptable in the lab, could be enhanced by adding fast embedded microcontrollers with error correction and detection software to the prototype area on the receive and transmit boards.

Bibliography

[1] Analog Devices. *Low Cost, Miniature Isolation Amplifiers.*

[2] Karl J. Astrom. *Computer Controlled Systems.* Prentice-Hall, Inc, Englewood Cliffs, N.J., 1984.

[3] S. Barnes and G. Smith. Automatic battery charging/discharging with a sine wave interface. *28th Universities Power Engineering Conference. Conference Proceedings.*, 1:42–5, September 1993.

[4] D. Beede, T. Curatolo, J. Dahlman, S. Girard, and G. Miller. A microprocessor based intelligent battery charger. *Proceedings of the Ninth Annual Battery Conference on Applications and Advances*, pages 185–90, January 1994.

[5] S. Biscaglia and D. Mayer. Modelling and analysis of lead acid battery operation. *Commission of the European Communities. Ninth E.C. Photovoltaic Solar Energy Conference. Proceedings of the International Conference.*, pages 245–8, September 1989.

[6] J.G. Bolger, C.A. Haslund, and R.J. Risser. Inductive charging of electric vehicles: testing and evaluation of an automated system. *Symposium Proceedings EVS-11. 11th International Electric Vehicle Symposium. Electric Vehicles: The Environment-Friendly Mobility*, 2(20.02):1–12, September 1992.

[7] A. Castelain, X. Fauvette, P. Le Moigne, and C. Rombaut. Battery charger with unity power factor for electric car. *Fifth International Conference on 'Power Electronics and Variable-Speed Drives'*, pages 568–73, October 1994.

[8] J. Cheema. Electric vehicles and other clean fuel alternatives: a comparative analysis. *Symposium Proceedings EVS-11. 11th International Electric Vehicle Symposium. Electric Vehicles: The Environment-Friendly Mobility*, 2(17.01):1–11, September 1992.

[9] F. Chenlo and J.B. Copetti. Lead/acid batteries for photovoltaic applications. test results and modelling. *Journal Power Sources*, 47:109–18, January 1994.

[10] J. Cohen. Fleet vans lead the way for electric vehicles. *EPRI Journal*, 11(5):22–9, July 1986.

[11] Claudio de Sa e Silva. *Power Factor Correction with the UC3854 - Application Note.* UNITRODE Integrated Circuits, 1990.

[12] D.M. Divan, K.W. Klontz, R.D. Lorenz, and D.W. Novonty. Contactless power delivery system for mining applications. *Conference Record of the 1991 IEEE Industry Applications Society Annual Meeting*, 2:1263–9, October 1991.

[13] Albert Esser. Contactless charging and communication system for electric vehicles. *Conference Record of the 1993 IEEE Industry Applications Conference Twenty-Eigth IAS Annual Meeting*, 2, October 1993.

[14] Yawen Gao. A bus system controlled 10 kw battery charger with igbt. *Symposium Proceedings EVS-11. 11th International Electric Vehicle Symposium. Electric Vehicles: The Environment-Friendly Mobility.*, 2(20.01):1–10, September 1992.

[15] Peter R. Gluck and Paul J. Timmerman. Modelling a nickel-cadmium battery as a heterogenous device. *Proceedings of the 25th Intersociety Energy Conversion Engineering Conference*, 2:27–31, August 1990.

[16] D.C. Hamill and E.K.G. James. Application of microprocessor technology to battery charging. *Power Sources and Supplies Conference Proceedings*, 2:1–9, January 1988.

[17] H. Hashimoto, H. Matsuki, K. Murakami, S. Nitta, M. Shiiki, and T. Yamamoto. *High efficient energy transmission for implantable artificial heart*, March 1993.

[18] D.J. Hind. Inductively coupled battery charging system. *7th Battery Conference. Conference Proceedings (ERA Report 92-0003).*, 3:1–13, April 1992.

[19] Emil W. Horst, Kornellis Klob, and Wim C. Turkenburg. Battery modelling for photovoltaic applications. *Eigth E.C. Photovoltaic Solar Energy Conference. Proceedings of the International Conference.*, 2:1564–8, May 1988.

[20] INTEL Corporation. *EV80C196KB Microcontroller Evaluation Board User's Mabual, Release 001*, February 1989.

[21] INTEL Corporation. *80C196KB User's Guide*, October 1990.

[22] M.G. Jayne and C. Morgan. The modelling of lead acid batteries for vehicle applications. *Proceedings of the 32nd International Power Sources Symposium*, pages 387–94, June 1986.

[23] G. Joos, Y. Lin, and J.F. Lindsay. Performance analysis of parallel - processing ups systems. *APEC '93. Eigth Annual Applied Power Electronics Conference and Exposition. Conference Proceedings.*, pages 533–9, March 1993.

[24] Edward W. Kamen. *Introduction to Signals and Systems.* Macmillan Publishing Company, New York, New York, second edition, 1990.

[25] John G. Kassakian, Martin F. Schlecht, and George C. Verghese. *Principles of Power Electronics.* Addison-Wesley, Reading, Massachusetts, 1991.

[26] P. Laffitte. The breakthrough of the electric vehicle: the necessary decisions that confront the public authorities. *Symposium Proceedings EVS-11. 11th International Electric Vehicle Symposium. Electric Vehicles: The Environment-Friendly Mobility*, 1(2.07):1–7, September 1992.

[27] S.B. Leeb. Recognition of dynamic patterns in high frequency dc-dc switching converters. Master's project, Massachusetts Institute of Technology, LEES, February 1989.

[28] S.B. Leeb, A.H. Mitwalli, G.C. Verghese, and V.J. Thottuvelil. A digital controller for a unity power factor converter. *IEEE Workshop on Computers in Power Electronics*, August 1992.

[29] M. Maja and P. Spinelli. A simplified model of the lead/acid battery. *Journal Power Sources*, 30:201–207, March 1990.

[30] Maxim. *Voltage-Output, 12-Bit Multiplying DACs - Data Sheet*, September 1990.

[31] Mike McVey and Deanna Temkin. A spacecraft electrical battery simulator. *Proceedings of the 25th Intersociety Energy Conversion Engineering Conference*, 2:19–26, August 1990.

[32] Micrometals. *Toroidal Cores - Data Sheet.*

[33] J.H. Mills. Inductran traction battery charging without cables. *Symposium Proceedings EVS-11. 11th International Electric Vehicle Symposium. Electric Vehicles: The Environment-Friendly Mobility*, 2(20.05):1–8, September 1992.

[34] Ahmed Mitwalli. A digital controller for a unity power factor converter. Master's project, Massachusetts Insititute of Technology, LEES, January 1993.

[35] Loveday H. Mweene, David M. Otten, and Martin F. Schlecht. A high-efficiency 1.5 kw, 390-50 v half-bridge converter operated at 100 *APEC 92. Seventh Annual Applied Power Electronics Conference and Exposition. Conference Proceedings 1992.*, pages 732–30, February 1992.

[36] Loveday H. Mweene, Martin F. Schlecht, and Chris A. Wright. A 1 kw, 500 khz front-end converter for a distributed power supply system. *APEC 89. Fourth Annual IEEE Applied Power and Electronics Conference and Exposition. Conference Proceedings 1989.*, pages 423–32, March 1989.

[37] Loveday Haachitaba Mweene. *The Design of Front-End DC-DC Converters of Distributed Power Supply Systems with Improved Efficiency and Stability.* PhD dissertation, Massachusetts Insititute of Technology, LEES, September 1992.

[38] H. Naganawa and M. Yamanoi. Optimum charging of a battery and its effects on charging efficiency. *Systems and Control*, 31(12):896–901, December 1987.

[39] National Semiconductor. *DS0026/DS0056 5 MHZ Two Phase MOS Clock Drivers - Data Sheet.*

[40] John A. O'Conner. *Unique Chip Pair Simplifies Isolated High Side Switch Drive - Application Note.* UNITRODE Integrated Circuits.

[41] A. Pourfallah. Advanced microprocessor control of battery charging devices for ni-cd and ni-mh batteries. *Power Quality '93 USA. Official Proceedings of the Seventh International Power Quality Conference*, pages 756–72, October 1993.

[42] S. Rahman and G.B. Shrestha. An investigation into the impact of electric vehicle load on the electric utility distribution system. *IEEE Transactions on Power Delivery*, 8(2):591–7, April 1993.

[43] Steven Shaw. High bandwidth connectorless communication in high frequency magnetics. *6.100 Independent Laboratory, Massachusetts Institute of Technology*, January 1994.

[44] William McC. Siebert. *Circuits, Signals, and Systems.* The MIT Press, Cambridge, MA, 1986.

[45] UNITRODE Integrated Circuits. *Linear Integrated Circuits High Power Factor Prereg-ulator - Preliminary*, 1990.

[46] P.M. Whittam. Charging management for vehicle batteries. *7th Battery Conference. Conference Proceedings (ERA Report 92-0003).*, 3(3):1–10, April 1992.